Minimizing Value-at-Risk in Single Machine Scheduling Problems

Semih Atakan

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

August, 2012

Minimizing Value-at-Risk in Single Machine Scheduling Problems

Approved by:

Assoc. Prof. Dr. Kerem Bülbül (Thesis Supervisor)	.....
Assist. Prof. Dr. Nilay Noyan Bülbül (Thesis Supervisor)	.....
Prof. Dr. Gündüz Ulusoy	.....
Assist. Prof. Dr. Kemal Kılıç	.....
Prof. Dr. Ali Rana Atılgan	.....

Date of Approval:

Acknowledgements

I would like to express my deepest gratitude to my thesis advisors Kerem Bülbül and Nilay Noyan for their huge and never-ending support. The ingenuity, the unyielding character, and the great deal of effort of Kerem Bülbül, and the skills, the knowledge and the guidance of Nilay Noyan were the factors that kept this research going. Without their hope, motivation, and assistance, this research would have never existed.

Secondly, I am grateful to my beloved fiancée, Birce, for her early contributions to this research, and her moral support through my thesis. She was always there for me, whenever I needed help.

I am thankful to many of our faculty members for their helpful advices and support. I am also thankful to my dear colleagues, Mahir, Çetin, Muzaffer, Halil, Nurşen, Mustafa, Belma and Ceyda, for sharing their experiences and their smile whenever necessary.

Finally, I would like to thank to my family for all the support they have provided. Without their wisdom, I would have never been able to earn a master's degree or write a thesis.

Tek Makinalı Çizelgeleme Problemlerinde Riske Maruz Değerin Enküçüklenmesi

Semih Atakan

Endüstri Mühendisliği, Yüksek Lisans Tezi, 2012

Tez Danışmanları: Kerem Bülbül, Nilay Noyan Bülbül

Anahtar Kelimeler: tek makinalı çizelgeleme; rassal işleme süreleri; rassal çizelgeleme; riske maruz değer; olasılıksal kısıt; rassal programlama; senaryo ayrışımı; kesi yaratma; eşlenik sabitleştirme; paralel programlama

Özet

Çizelgeleme literatürünün büyük bir çoğunluğu tüm verinin önceden bilindiği belirlenimci problemlere odaklanır. Bu varsayım, problem parametrelerindeki değişkenlik seviyesinin düşük olduğu durumlar için mantıklı olabilir; ancak değişkenlik seviyesi arttıkça oluşabilecek kötü sonuçları engellemek için belirsizliğin modele dahil edilmesi büyük önem taşımaktadır. Bu tezde, belirsiz problem parametreleri içeren tek makinalı çizelgeleme problemleri incelenmektedir. Rassal bir performans ölçütüne (örneğin tamamlanma süresi, ağırlıklı tamamlanma süresi, ağırlıklı gecikme süresi) ilişkin bir olasılıksal kısıt tanımlanarak riskten kaçınan genel bir rassal programlama modeli önerilmektedir. Bu modelin hedefi, rassal performans ölçütüne ilişkin belli bir güven seviyesindeki riske maruz değeri (VaR) enküçükleyen, statik ve kesinti içermeyen bir görev işleme sırası bulmaktır. Bu çalışmada en iyi VaR değeri için sıkı üst ve alt sınırlar bulabilmek amacıyla Lagrange gevşetmesini temel alan bir ayrıştırma stratejisi izlenmektedir. Lagrange eşleniği problemini çözmek için sabitleştirilmiş bir kesi yaratma algoritması geliştirilmiştir. Ayrıca, önerilen modelin ve çözüm yöntemlerinin önemini göstermek amacıyla, üç rassal performans ölçütü kullanarak sayısal analiz yapılmıştır.

Minimizing Value-at-Risk in Single Machine Scheduling Problems

Semih Atakan

Industrial Engineering, Master's Thesis, 2012

Thesis Supervisors: Kerem Bülbül, Nilay Noyan Bülbül

Keywords: single-machine scheduling; stochastic processing times; stochastic scheduling; value-at-risk; probabilistic constraint; stochastic programming; scenario decomposition; cut-generation; dual stabilization; parallel programming

Abstract

The vast majority of the machine scheduling literature focuses on deterministic problems in which all data is known with certainty a priori. This may be a reasonable assumption when the variability in the problem parameters is low. However, as variability in the parameters increases incorporating this uncertainty explicitly into a scheduling model is essential to mitigate the resulting adverse effects. In this thesis, we consider single-machine scheduling problems in the presence of uncertain problem parameters. We impose a probabilistic constraint on the random performance measure of interest (such as the total completion time, total weighted completion time, and total weighted tardiness), and introduce a generic risk-averse stochastic programming model. In particular, the objective of the proposed model is to find a non-preemptive static job processing sequence that minimizes the value-at-risk (VaR) of the random performance measure at a specified confidence level. In this study, we propose a Lagrangian relaxation based decomposition strategy to obtain tight lower and upper bounds for the optimal VaR. In order to solve the Lagrangian dual problem we provide a stabilized cut-generation algorithm. We also present an extensive computational study on three selected performance measures to demonstrate the effectiveness of our solution methods and the value of the proposed model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the scheduling literature, many objectives are proposed for different production environments. Among all of them, one of the most common objectives is to minimize the total completion times (TCT) of jobs at hand. This objective could be extended by assigning unit weights to jobs where the weights would represent the jobs' importance or urgency. As a result, in an optimal job sequence, the jobs with higher weights will more likely to be processed at earlier stages. Such an objective is called the minimization of the total weighted completion time (TWCT). In the single-machine scheduling literature, both of these objectives are considered as easy problems due to their special structures. A more difficult problem has the total weighted tardiness (TWT) objective which is a due date related performance measure in make-to-order environments. The goal is to find a job (order) processing sequence in order to minimize the total cost incurred due to missed due dates. For a given job, the cost is directly proportional to the associated tardiness. The unit tardiness cost (weight) may either be associated with the perceived penalty due to a loss of customer goodwill or may represent actual contractual penalties. The interested reader is referred to Sen et al. (2003) for a recent survey on the topic.

In the traditional single-machine problems described above, all processing times, release dates, due dates, and weights are known in advance at time zero with certainty. However, in many practical settings the exact values of one or several of these parameter types may not be available at the time the dispatcher determines a job processing sequence. In particular, possible machine breakdowns, variable setup times, inconsistency of the worker performance, or changes in tool quality may introduce uncertainty into the processing times. The uncertainty in the processing time of a job is resolved at the time of the job completion. The models developed in this thesis can be generalized to incorporate

randomness into all parameters. However, from a practical point of view it is reasonable to presume that a due date is quoted as a result of a mutual agreement with the customer, and the unit tardiness weight associated with a customer is also known based on either the internal priority of the customer or the contractual agreement. Therefore, in our computational experiments the due dates and the unit weights are deterministic. Furthermore, we assume that all jobs are ready to be released at time zero. Consequently, we focus on the uncertainty in the processing times which leads to uncertain completion times and tardiness values. Our objective is to determine a risk-averse fixed job processing sequence at time zero that hedges against the uncertainty in the processing times. In the stochastic scheduling terminology (see Pinedo (2008)), we construct a non-preemptive static list policy.

Traditional models for decision making under uncertainty define optimality criteria based on expected values and disregard variability inherent in the system. Following this mainstream risk-neutral approach, most of the classical stochastic scheduling puts a lot of effort into analyzing the expected performance by assuming that uncertain parameters such as processing times follow specific distributions. See Pinedo (2008) for an excellent overview of conventional stochastic scheduling. However, variability typically implies a deterioration in performance, and risk-neutral models may provide solutions that perform poorly under certain realizations of the random data. Capturing the effect of variability can be accomplished by incorporating the appropriate risk measures into the model that reflect the preferences of the decision maker. Several criteria to select risk measures have been discussed in the literature (see, e.g., Ogryczak and Ruszczyński (1999, 2002); Artzner et al. (1999)). Considering the wide range of criteria, there is no universally accepted single risk measure appropriate for all decision making contexts. In this study, we consider the VaR measure which is a very popular and widely applied risk measure in the finance literature. For the studies related to VaR we refer to the chapter by Larsen et al. (2002). In our context, we focus on either the TCT, or the TWCT, or the TWT as the random outcome associated with a fixed job processing sequence selected at time zero. The goal is to specify the smallest possible upper bound on the random performance measure that will be exceeded with at most a pre-specified small probability. Here, the selected upper bound is the VaR of the random performance measure at the desired probability level, and we minimize VaR. The concept of VaR is closely related to probabilistic constraints. Stochastic programming models with probabilistic constraints were introduced by Charnes et al. (1958) and have been employed successfully in a variety of fields. The interested reader can refer to Prékopa (1995) and Dentcheva (2006) for reviews and a

comprehensive list of references. Our proposed approach is an intuitive and practical way of modeling a service level requirement for the performance measure under the stochastic setup and leads to a novel risk-averse stochastic programming model. To the best of our knowledge, this is a first in the machine scheduling literature.

It is well known that models incorporating VaR exhibit a non-convex structure even if the underlying deterministic problem is convex. The existing solution methods primarily deal with VaR integrated into a linear program (LP). Thus, the decision variables are continuous, and VaR is introduced on a random outcome expressed as a linear function of the decision variables. Larsen et al. (2002) provide a review of the algorithms available for solving such problems. Note that these studies are generally concerned with portfolio optimization problems. Larsen et al. (2002) also introduce two heuristic algorithms which solve a series of problems involving a related risk measure known as conditional-value-at-risk (CVaR). In contrast to VaR, the problem of minimizing CVaR can be formulated as an LP if the uncertainty is represented by a set of scenarios, and the proposed heuristics use LP techniques iteratively. However, in our study the underlying problem involves sequencing decisions that can only be expressed by employing binary variables; and therefore, even minimizing CVaR is hard. Consequently, the proposed solution methods do not apply in our case.

We characterize the randomness associated with the uncertain parameters by a finite set of scenarios, where a scenario represents a joint realization of all random parameters. It is important to point out that the scenario approach allows us to generate data from any distribution and, for instance, to model the correlation of the random processing times among different jobs by considering their joint realizations. In this sense, a scenario-based approach is more general than assuming specific distributions. On the down side, the computational complexity of solving the problem is closely affected by the number of scenarios. There are only a few studies utilizing a scenario-based approach for machine scheduling problems. For example, Gutjahr et al. (1999) minimize the expected TWT with stochastic processing times and propose a stochastic branch-and-bound technique, where a sampling approach is embedded into their bounding schemes. Alternatively, other existing scenario-based studies develop robust optimization models in order to optimize the worst-case performance over all scenarios. Such a worst-case analysis does not require the probabilities of the scenarios. The sum of completion times is employed in Daniels and Kouvelis (1995); Yang and Yu (2002), and the weighted sum of completion times is considered by de Farias et al. (2010), while Kasperski (2005) focuses on the maximum lateness as the random performance criterion. One or several of the robustness measures

known as the maximum deviation from optimality, the maximum relative deviation from optimality, and the maximum value over all scenarios are incorporated in these papers. Except de Farias et al. (2010), all these studies design specialized algorithms for the robustness measure and random performance criterion of interest. de Farias et al. (2010) identify a family of valid inequalities to strengthen the mixed-integer formulation of their problem. Furthermore, Alouloua and Croce (2008) provide several complexity results in the domain of robust scheduling. In contrast to robust approaches adopting a conservative worst-case view, we define our optimality criterion based on VaR which is a quantile of the random outcome at a specified probability level. That is, we utilize probabilistic information and develop a risk-averse stochastic programming model alternative to existing robust optimization models. Note that setting the required probability level to exactly one, subsumes the robust optimization problem of minimizing the maximum performance measure over all scenarios. However, when the required probability level is specified as $\alpha < 1$, we minimize the maximum performance measure over a subset of scenarios with an aggregate probability of at least α. Our risk-averse model identifies the optimal subset of scenarios with the specified minimum aggregate probability level and minimizes the maximum performance measure over this subset. Thus, it is less conservative than the robustness approach which considers all scenarios.

The major contribution of this study is to develop a risk-averse model that is novel in machine scheduling. We analyze the behavior of the proposed model in comparison to that of the risk-neutral model and provide insights on the impact of the risk preference. Furthermore, in all papers on robust scheduling mentioned above the corresponding deterministic single-machine problems are polynomially solvable. In our study, the TCT and TWCT objectives are polynomially solvable too. However, the single-machine TWT problem is strongly $\mathcal{NP}$-hard (Lenstra et al. (1977)), and incorporating VaR poses additional computational difficulties.

Not limited to VaR, stochastic programming models are generally known to be computationally challenging. This can be partially attributed to the potentially large number of scenario-dependent variables and constraints. Various decomposition based solution methods have been proposed to deal with such large scale programs. For example, the L-shaped method proposed by Van Slyke and Wets (1969) is a widely applied Benders-decomposition approach to solve the two-stage linear stochastic programming problems with the expected recourse functions for the case of a finite probability space. Such L-shaped algorithms are based on a cutting plane algorithm, where the cuts are constructed using the dual information of the second-stage problems associated with each scenario.

However, when the second-stage problems involve integer variables, the standard decomposition methods utilizing the linear programming duality cannot be applied. Introducing integer variables into linear stochastic programs further complicates solving these models.

In the stochastic programming literature, the studies that focus on developing solution methods for such integer programs mainly consider the two-stage framework. The integer L-shaped decomposition algorithm proposed by Laporte and Louveaux (1993) is the first one that uses a decomposition method for stochastic programs with integer decisions in the second-stage. It utilizes a branch-and-cut scheme in the master problem and it is proposed only for the case of pure binary first-stage variables. Carøe and Tind (1998) generalize the integer L-shaped algorithm for the models with mixed-integer first- and second-stage variables. They use general duality theory to approximate the second-stage value function in the space of the first-stage variables and obtain non-linear cuts. However, there is no practical method for solving the resulting master problem as emphasized in Ahmed et al. (2004).

Alternatively, Carøe and Schultz (1999) use the scenario decomposition approach of Rockafellar and Wets (1991) and develop a branch-and-bound algorithm based on the Lagrangian relaxation of non-anticipativity. Recently, this solution approach has been adapted to two-stage stochastic integer programs incorporating risk measures such as excess probabilities (Schultz and Tiedemann, 2003) and CVaR (Schultz and Tiedemann, 2006). In this thesis, we adapt their Lagrangian-relaxation based decomposition approach, which is originally developed for two-stage models, to our single-stage stochastic integer programming model. For a detailed discussion on various algorithms for stochastic integer programming we refer the reader to Birge and Louveaux (1997), Klein Haneveld and van der Vlerk (1999), and Louveaux and Schultz (2003). In order to solve the Lagrangian dual problem, we propose a cut generation algorithm which is enhanced with dual stabilization methods to achieve faster convergence. We also utilize parallel programming techniques in order to improve the performance of our algorithm. We note that our proposed solution method is not limited to machine scheduling but could be applied to a wide variety of problems.

In the next chapter, we formally define the risk-averse scheduling problems and present their mathematical programming formulations. In Chapter 3, we introduce our solution strategy and discuss the implementation details of the proposed cut-generation algorithm. Computational results are presented in Chapter 4, and we conclude in Chapter 5 with further research directions.

Chapter 2

Modeling Value-at-Risk

In this section, we first present the underlying deterministic model of the stochastic single-machine scheduling problem that we are focusing on. Then, we discuss how to model the uncertainty inherent in the system and develop our risk-averse stochastic programming model.

2.1 Underlying Deterministic Single Machine Scheduling Model

A machine scheduling problem can be considered as a two-phase optimization problem. In the first phase, a feasible job processing sequence is determined for each machine involved, and then in the second phase the optimal start and completion times are computed for fixed job processing sequences. The difficult combinatorial structure of machine scheduling problems stems from the first phase, while the second phase - also referred to as the *optimal timing problem* - is a simple optimization problem for many important machine scheduling problems. On a single machine, the optimal timing problem is trivial for regular objectives, and it can often be solved by a low-order polynomial time algorithm or as a linear programming problem for non-regular objectives. Since our focus is on regular objectives, we will not require custom optimal timing algorithms in our work.

For single-machine scheduling problems, four frequently used alternate deterministic formulations appear in the literature (see Keha et al. (2009)): disjunctive (DF), time-indexed (TIF), linear ordering (LOF), and the assignment and positional date formulations (APDF). TIF has a tight LP relaxation and is the best contender among these four formulations if the processing times are small (Keha et al. (2009)). TIF, however, can-

not be adapted to our stochastic setting directly, because it infers the sequence from the completion times represented by binary decision variables. Recall that our goal is to find a non-preemptive static job processing sequence at time zero. That is, the decisions are independent of the random realizations of data, and therefore, relying on completion time information that is contingent on the random processing times (and random release dates if applicable) is not appropriate to construct a static job processing sequence. Our preliminary results indicate that DF is outperformed by LOF and APDF in terms of computational time. This observation is also supported by the extensive computational study presented in Keha et al. (2009). Thus, among the common formulations only LOF and APDF are viable options for our proposed risk-averse model. In this study, we work with both of these formulations in order to exploit their structural properties for different objective functions.

We define the set of jobs to be processed as $N := \{1, \ldots, n\}$, where n denotes the number of jobs. Associated with each job $j \in N$ are several parameters: a processing time p_j, a due date d_j, and a tardiness cost or a completion time penalty per unit time w_j depending on the objective function used. In the APDF formulation presented next, the binary variable x_{jk} takes the value 1 if job j is assigned to position k in the sequence, and is set to zero otherwise. Assuming zero release dates, a deterministic single-machine scheduling problem, described as $1//f(\boldsymbol{x})$ following the common three field notation of Graham et al. (1979), is formulated below:

$$\begin{aligned}
\min \quad & f(\boldsymbol{x}) \\
\text{subject to} \quad & \sum_{k \in N} x_{jk} = 1, && \forall j \in N, && (2.1)\\
& \sum_{j \in N} x_{jk} = 1, && \forall k \in N, && (2.2)\\
& x_{jk} \in \{0, 1\} && \forall j, k \in N. && (2.3)
\end{aligned}$$

The constraints (2.1)-(2.2) ensure that job j is placed to exactly one position and the position k is used exactly by a single job. Constraints (2.3) are the integrality restrictions. If we are only interested in the TCT, we can express the objective function as:

$$\sum_{j \in N} C_j = \sum_{j \in N} \sum_{k=1}^{n} (n - k + 1) p_j x_{jk}, \tag{2.4}$$

where C_j is the completion time of job j. Notice that, for other objective functions such

as TWCT or TWT we must know the individual C_j's. This requires additional constraints and variables which are discussed in Keha et al. (2009).

The LOF of single machine scheduling problems uses a binary variable δ_{jk} which takes the value 1, if job j precedes job k in the processing sequence, and is zero otherwise. By convention, we set $\delta_{jj} = 1$ for all $j \in N$. The formulation is presented below:

$$
\begin{aligned}
\min \quad & f(\boldsymbol{\delta}) \\
\text{subject to} \quad & \delta_{jj} = 1, && \forall j \in N && (2.5)\\
& \delta_{jk} + \delta_{kj} = 1, && 1 \le j < k \le n, && (2.6)\\
& \delta_{jk} + \delta_{kl} + \delta_{lj} \le 2, \qquad \forall j,\ k,\ l \in N\ : && j \neq k,\ k \neq l,\ l \neq j, && (2.7)\\
& \delta_{jk} \in \{0, 1\} && \forall j, k \in N. && (2.8)
\end{aligned}
$$

Constraints (2.6) ensure that for each pair of jobs j and k either job j precedes job k or vice versa. Constraints (2.7) represent the transitivity requirements for a linear ordering of the jobs. In other words, they guarantee that for any triplet of jobs j, k, l, if job j precedes job k and job k precedes job l then job j precedes job l. Constraints (2.8) are the binary variable restrictions required for the sequencing decisions. The completion time of job j is the sum of the processing times of all of its predecessors $C_j = \sum_{k \in N} \delta_{kj} p_k$, (recall that $\delta_{jj} = 1$ by convention). Thus, the LOF for minimizing TWCT on a single-machine is stated as:

$$
\begin{aligned}
\min \quad & \sum_{j \in N} w_j C_j \\
\text{subject to} \quad & (2.5) - (2.8).
\end{aligned}
$$

The tardiness T_j of job j is expressed by $T_j = \max(0, C_j - d_j)$. Due to its structure, in order to model due date related performance measures, we require the following set of constraints:

$$
\begin{aligned}
& T_j \ge C_j - d_j && \forall j \in N && (2.9)\\
& T_j \ge 0 && \forall j \in N. && (2.10)
\end{aligned}
$$

The LOF for minimizing TWT can now be stated as:

$$
\min \quad \sum_{j \in N} w_j T_j
$$

$$\text{subject to} \quad (2.5) - (2.10).$$

In the remainder of the thesis, we will use the APDF in order to decribe our risk-averse model and our solution approach. Note that the discussion will be general and applies to both APDF and LOF. When necessary, the modifications to use LOF will also be provided.

2.2 Risk-Averse Stochastic Programming Model

In our setting, the actual values of the processing times are not certain at the time we determine the job processing sequence, and the processing times can be represented by random variables. This implies that the completion times $C(\boldsymbol{x})$ and the tardiness values $T(\boldsymbol{x})$ associated with a sequence are also random variables, since they are functions of the random processing times. In this case, comparing alternate candidate sequences requires comparing their respective random $f(\boldsymbol{x})$ values. We propose a risk-averse approach which evaluates a sequence with respect to a certain quantile of the distribution of the associated random $f(\boldsymbol{x})$. Let Υ and ξ_j denote the random $f(\boldsymbol{x})$ and the random processing time of job $j \in N$, respectively. The random variable Υ is a random outcome associated with a sequence $\boldsymbol{x} \in \{0,1\}^{n \times n}$. Using the expression in (2.4), we can represent Υ as a function of the decision vector $\boldsymbol{x} \in \{0,1\}^{n \times n}$ for the TCT objective as follows:

$$\Upsilon = \sum_{j \in N} \sum_{k=1}^{n} (n-k+1) \xi_j x_{jk}. \tag{2.11}$$

Similarly, using the decision vector $\boldsymbol{\delta} \in \{0,1\}^{n \times n}$, the random TWT is expressed as:

$$\Upsilon = \sum_{j=1}^{n} w_j \max \left(\sum_{k=1}^{n} \xi_k \delta_{kj} - d_j, 0 \right). \tag{2.12}$$

We intend to model the risk associated with the variability of the random outcome Υ by introducing the following probabilistic constraint:

$$P(\Upsilon \leq \theta) \geq \alpha, \tag{2.13}$$

where α is a specified large probability such as 0.90 or 0.95. Here θ denotes an upper bound on the $f(\boldsymbol{x})$ that is exceeded with at most a small probability of $1-\alpha$. If $\alpha = 1$, $\Upsilon \leq \theta$ holds almost surely. As discussed in more depth in Chapter 1, such a probabilistic

constraint is intuitive and allows us to model a service level requirement for the $f(\boldsymbol{x})$ under the stochastic setup. We refer to α as the risk parameter which reflects the level of risk-aversion of the decision maker. Clearly, increasing α results in allowing a higher value of the upper bound θ. We propose not to specify the value of θ as an input, but consider it as a decision variable with the purpose of identifying the sequence with the smallest possible value of θ given the risk aversion of the decision maker. Thus, in our model we minimize θ for a specified parameter α, which is equivalent to minimizing the α-quantile of the random $f(\boldsymbol{x})$. The α-quantile has a special name in risk theory as presented in the next definition.

Definition 1 *Let X be a random variable. The α-quantile*

$$\inf\{\eta \in \mathbb{R} : \ F_X(\eta) \geq \alpha\}$$

is called the Value at Risk (VaR) at the confidence level α and denoted by VaR$_\alpha(X)$, $\alpha \in (0, 1]$.

Figure 2.1 visualizes the concept of VaR associated with the random TWT using an instance from our computational study.

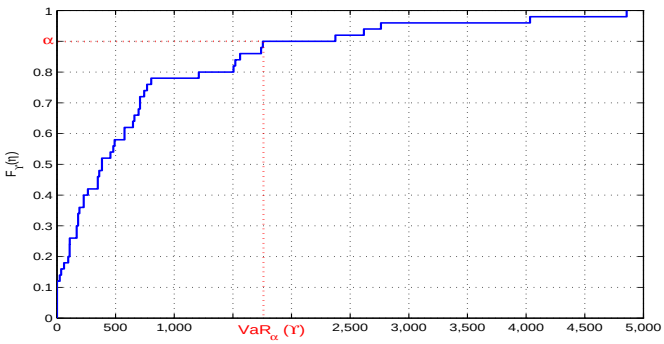


Figure 2.1: The VaR$_\alpha(\Upsilon)$ associated with the best feasible sequence obtained for an instance from our computational study.

The probabilistic constraint (2.13) can equivalently be formulated as a constraint on the VaR of the random $f(\boldsymbol{x})$:

$$\text{VaR}_\alpha(\Upsilon) \leq \theta. \tag{2.14}$$

In other words, by considering the proposed probabilistic constraint (2.13) we specify the VaR as the risk measure on the random $f(\boldsymbol{x})$, and minimizing θ corresponds to seeking the sequence with the smallest possible VaR value for a specified α value.

A model with a probabilistic constraint similar to that in (2.13) with randomness on the left hand side was first studied by de Panne and Popp (1963) and Kataoka (1963). Kataoka introduces a transportation type model and Van de Panne and Popp present a diet (cattle feed) optimization model with a single probabilistic constraint. In these studies, the random outcome of interest is a linear function of the decision vector, and in both studies the solution methods are specific to random coefficients with a joint normal distribution. In contrast, the random outcome Υ in our work is not a linear function of the decision vector as evident from (2.12), and we do not assume that it has a specific distribution.

We characterize the random processing times by a finite set of scenarios denoted by S, where a scenario represents a joint realization of the processing times of all jobs. To develop our stochastic programming formulation, previously introduced parameters and variables are augmented with scenario indices and a probability vector $\boldsymbol{\pi}$ is added:

π^s: probability of scenario $s,\ s \in S$.

p_j^s: processing time of job j under scenario $s,\ s \in S$.

C_j^s: completion time of job j under scenario $s,\ s \in S$.

T_j^s: tardiness of job j under scenario $s,\ s \in S$.

Then, using APDF we formulate the problem of minimizing the VaR in the single machine scheduling problem as follows:

$$\begin{aligned}
\min \quad & \theta & & (2.15)\\
\text{subject to} \quad & \sum_{k\in N} x_{jk} = 1, & \forall j \in N, \quad & (2.16)\\
& \sum_{j\in N} x_{jk} = 1, & \forall k \in N, \quad & (2.17)\\
& f^s(\boldsymbol{x}) - \theta \le f^s_{\max}\beta^s, & \forall s \in S, \quad & (2.18)\\
& \sum_{s\in S} \pi^s\beta^s \le 1-\alpha, & & (2.19)\\
& \beta^s \in \{0,1\}, & \forall s \in S, \quad & (2.20)\\
& x_{jk} \in \{0,1\}, & \forall j,k \in N, \quad & (2.21)\\
& \theta_{LB} \le \theta \le \theta_{UB}. & & (2.22)
\end{aligned}$$

We emphasize that the constraints (2.16), (2.17) and (2.21) in the model above are identical to the constraints (2.1)-(2.3). That is, the sequencing decisions are independent of

the uncertainty. The constraints (2.18)-(2.20) represents the probabilistic constraint in (2.14). The parameter $f^s_{\max}$ stands for a valid upper bound on $f^s(\boldsymbol{x})$ for any sequence $\boldsymbol{x}$ under scenario s. This parameters guarantees that the binary variable β^s is set to 1 by the corresponding constraint (2.19) if $f^s(\boldsymbol{x})$ exceeds the threshold value θ in scenario s. Constraint (2.19) mandates that the probability of exceeding the threshold value θ for the random outcome is no more than $1-\alpha$. For the validity of the formulation (2.16)-(2.22), we must ensure that $f^s_{\max}$ is no smaller than the maximum possible $f^s(x)$ under scenario s. In order to obtain a reasonably tight formulation, we sort the processing times under scenario s in non-increasing order and denote the jth largest processing time under scenario s by $p^s_{[j]}$. Then, the maximum possible completion time of the kth job in the sequence, $k \in N$, under scenario s is computed as $C^s_{[k]} = \sum_{j=1}^{k} p^s_{[j]}$. Next, the due dates and the unit tardiness or processing weights are assigned to the completion times in non-increasing and non-decreasing order, respectively. A standard pairwise interchange argument (not necessarily adjacent) demonstrates that the resulting TWT is an upper bound on the TWT of any job processing sequence under scenario s. Similar bounds can easily be computed for the other objectives of interest as well.

The final constraint (2.22) is incorporated in order to improve the convergence of our proposed algorithm in Section 3.4.7. While, θ_{UB} could be set to the VaR of any feasible sequence of jobs, in the absence of a valid θ_{LB}, one can simply use 0 in its place. The LOF for minimizing VaR is the same as (2.15)-(2.22) once (2.16), (2.17) and (2.21) are replaced by their counterparts (2.5)-(2.8). In order to calculate the resulting job tardiness values, the tardiness constrains (2.9)-(2.10) should be duplicated for every scenario and appended to the formulation above. At an optimal solution, T^s_j may be strictly larger than $\max\{C^s_j - d_j, 0\}$ for some scenario $s \in S$ because the tardiness values are not associated with positive cost coefficients in the objective. Obviously, we preserve optimality by setting $T^s_j = \max\{C^s_j - d_j, 0\}$.

Uncertainty in the due dates and/or the unit tardiness or processing costs may be incorporated in our formulation in a straightforward manner by replacing the parameters d_j and w_j by d^s_j and w^s_j while calculating the $f^s(x)$. This modification does not affect the number of variables and constraints. However, if the release dates are not known in advance, then the completion time expression must be replaced by a set of constraints which is adapted from the deterministic formulation in Nemhauser and Savelsbergh (1992):

$$C^s_j \geq r^s_i \delta_{ij} + \sum_{\{k\,:\, r^s_k < r^s_i,\ k \neq j\}} p^s_k(\delta_{ik} + \delta_{kj} - 1) + \sum_{\{k\,:\, r^s_k \geq r^s_i\}} p^s_k \delta_{kj}, \quad \forall i, j \in N.$$

Notice that the constraints above are for LOF. In the remainder of the thesis, we refer to the formulation (2.15)-(2.22) as VaR-$f(x)$.

Chapter 3

Solution Methods

As discussed in Chapter 1, several decomposition based solution methods have been offered to solve stochastic programming models but mainly for the two-stage stochastic integer programs. Among these existing methods, we utilize the one proposed by Carøe and Schultz (1999) for the stochastic models with mixed-integer first and second-stage variables. They consider a scenario decomposition approach and develop a branch-and-bound algorithm based on the Lagrangian relaxation of non-anticipativity. We adapt their approach to obtain a Lagrangian relaxation based decomposition to obtain tight lower and upper bounds for the optimal objective value of our single-stage stochastic integer programming model. In particular, we consider a split-variable formulation which is essentially based on the idea of creating copies of variables and then relaxing the constraints that force all these variables to be equal. This idea has been introduced in combinatorial optimization as variable splitting by Jörnsten et al. (1985). In studies that focus on two-stage models (Carøe and Schultz, 1999; Schultz and Tiedemann, 2003), the non-anticipativity conditions state that the first-stage decision should not depend on the scenario which will prevail in the second stage. In our single-stage setting they guarantee that the static job sequence decisions should not depend on the scenario. We note that our proposed solution method is not limited to the machine scheduling problem of interest. To the best of our knowledge, considering such a variable splitting based Lagrangian relaxation algorithm for minimizing VaR is the first in the literature.

In the following sections, we will present this Lagrangian relaxation based decomposition strategy. Then, we will present a method to provide upper and lower bounds on the optimal VaR measure which is used as an initialization to our solution approach. Next, we will discuss our solution methods for the Lagrangian problems, and finally introduce

the stabilized *cut-generation algorithm* to solve the Lagrangian dual problem.

3.1 Scenario Decomposition Using Lagrangian Relaxation

In order to carry out the decomposition, we create copies of the variables θ, x_{jk} and $\delta_{jk},\ \forall j,k \in N$, for each scenario. Accordingly, θ is replaced by θ^s in constraints (2.18), the constraints (2.16), (2.17) and (2.21) are replicated for each scenario, and the following non-anticipativity constraints are appended to the formulation (2.16)-(2.22):

$$(1-\pi^1)x^1_{jk} = \sum_{s=2}^{|S|} \pi^s x^s_{jk} \qquad \forall j,k \in N \tag{3.1}$$

$$\theta^1 = \theta^s \qquad \forall s \in S, s \neq 1. \tag{3.2}$$

Note that the non-anticipativity constraints (3.1) are valid because the variables $x_{jk},\ \forall j,k \in N$, are binary. The objective term θ in (2.15) is replaced by the equivalent expression $\sum_{s\in S} \pi^s\theta^s$ based on (3.2) and $\sum_{s\in S}\pi^s = 1$. In addition, note that

$$(1-\alpha) = \sum_{s\in S} \pi^s(1-\alpha), \tag{3.3}$$

and the term $(1-\alpha)$ on the right hand side of (2.19) is substituted accordingly. The resulting model is presented below.

$$\begin{aligned}
& \min \quad \sum_{s\in S} \pi^s\theta^s \\
\text{subject to} \quad & \sum_{k\in N} x^s_{jk} = 1, && \forall j \in N, s \in S \\
& \sum_{j\in N} x^s_{jk} = 1, && \forall k \in N, s \in S \\
& f^s(\boldsymbol{x}) - \theta^s \leq f^s_{\max}\beta^s, && \forall s \in S, \\
& \sum_{s\in S} \pi^s\beta^s \leq \sum_{s\in S} \pi^s(1-\alpha), \\
& \beta^s \in 0,1, && \forall s \in S, \\
& x^s_{jk} \in 0,1, && \forall j,k \in N, s \in S \\
& \theta_{LB} \leq \theta^s \leq \theta_{UB},
\end{aligned}$$

$$(1-\pi^1)x^1_{jk} = \sum_{s=2}^{|S|} \pi^s x^s_{jk}, \qquad \forall j,k \in N,$$
$$\theta^1 = \theta^s \qquad \forall s \in S, s \neq 1.$$

Notice that this model is exactly the same as (2.15)-(2.22) due to the appended nonanticipativity constraints. The Lagrangian $L(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is then obtained by dualizing the constraint (2.19) by a non-negative multiplier λ, and the constraints (3.1) and (3.2) by unrestricted multipliers $u_{jk},\ \forall j,k \in N$, and $\mu^s,\ s = 2,\ldots,|S|$, respectively:

$$\begin{aligned} L(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = &\sum_{s=2}^{|S|} \mu^s \pi^s (\theta^s - \theta^1) + \lambda \sum_{s\in S} \pi^s \left(\beta^s - 1 + \alpha\right) \\ &+ \sum_{j\in N}\sum_{k\in N} u_{jk} \left(\sum_{s=2}^{|S|} \pi^s x^s_{jk} - (1-\pi^1)x^1_{jk} \right). \end{aligned} \tag{3.4}$$

or in a more compact form:

$$L(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \sum_{s\in S} \pi^s\theta^s + \lambda \sum_{s\in S} \pi^s(\beta^s - 1 + \alpha) + \sum_{s\in S} \mu^s\theta^s + \sum_{j\in N}\sum_{k\in N}\sum_{s\in S} u_{jk}\boldsymbol{H}^s x^s_{jk}, \tag{3.5}$$

where

$$\mu^1 = -\sum_{s=2}^{|S|} \mu^s, \tag{3.6}$$
$$\boldsymbol{H} = \begin{bmatrix} (\pi^1 - 1) & \pi^2 & \pi^3 & \cdots & \pi^{|S|} \end{bmatrix}, \text{ and} \tag{3.7}$$

$\boldsymbol{H}^s$ represents the sth component of the vector $\boldsymbol{H}$ defined in (3.7). As a result, the Lagrangian decomposes for each scenario:

$$L^s(\lambda, \mu^s, \boldsymbol{u}) = (\pi^s + \mu^s)\theta^s + \lambda\pi^s(\beta^s - 1 + \alpha) + \sum_{j\in N}\sum_{k\in N} u_{jk}\boldsymbol{H}^s x^s_{jk}, \tag{3.8}$$
$$L(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \sum_{s\in S} L^s(\lambda, \mu^s, \boldsymbol{u}). \tag{3.9}$$

Note that μ^1 is only defined for notational convenience and is not a component of $\boldsymbol{\mu} = [\mu^2\ \mu^3\ \ldots\ \mu^{|S|}]$.

The analysis above provides us with $|S|$-many minimization problems, and for fixed

$\lambda, \boldsymbol{\mu}, \boldsymbol{u}$, the Lagrangian subproblems are defined as:

$$D(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \min_{\boldsymbol{x},\boldsymbol{\beta},\boldsymbol{\theta}} \quad \sum_{s \in S} L^s(\lambda, \mu^s, \boldsymbol{u}). \tag{3.10}$$

Here, $D(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is called the dual function. Our goal is to find the maximum value that D can take which we achieve by solving the Lagrangian dual problem:

$$\max_{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}} \quad D(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \max_{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}} \quad \sum_{s \in S} D^s(\lambda, \mu^s, \boldsymbol{u}), \tag{3.11}$$

where

$$D^s(\lambda, \mu^s, \boldsymbol{u}) = \min_{\boldsymbol{x},\beta^s,\theta^s} \quad L^s(\lambda, \mu^s, \boldsymbol{u}) \tag{3.12}$$

$$\text{subject to} \quad \sum_{k \in N} x^s_{jk} = 1, \qquad \forall j \in N, \tag{3.13}$$

$$\sum_{j \in N} x^s_{jk} = 1, \qquad \forall k \in N, \tag{3.14}$$

$$f^s(\boldsymbol{x}^s) - \theta \leq f^s_{\max} \beta^s, \tag{3.15}$$

$$\beta^s \in \{0, 1\}, \tag{3.16}$$

$$x_{jk} \in \{0, 1\}, \qquad \forall j, k \in N, \tag{3.17}$$

$$\theta_{LB} \leq \theta \leq \theta_{UB}. \tag{3.18}$$

Note that the dual function is non-differentiable and non-smooth. Therefore, we have to employ methods from nondifferentiable optimization in order to solve the Lagrangian dual problem.

To formulate this problem using LOF, one must replace $\boldsymbol{x}$ with $\boldsymbol{\delta}$ and substitute constraints (3.13)-(3.14) with the replicated versions of their counterparts described in (2.5)-(2.7). Unfortunately, the structure of the Lagrangian subproblems (3.12)-(3.18) formulated using either APDF or LOF do not seem amenable to an efficient solution procedure. Therefore, we will be tackling the subproblems using an integer programming solver as described in Section 3.3.

3.2 Bounding the Value-at-Risk

As discussed earlier, in order to improve the quality of our solutions, we impose a lower and an upper bound on θ^s. Notice that adding more constraints to our subproblems reduces the feasible region. Within a more restricted feasible region, the optimal solution of the new subproblem will be greater or equal to the optimal solution of the original subproblem. In return, the optimal objective function value of the Lagrangian dual problem will be greater than or equal to the original Lagrangian dual problem's objective function value. This means that by adding bounds on θ^s, we can actually improve the quality of our solutions. Below, we provide a method to generate tight bounds for the optimal VaR value as a preprocessing method.

The relation of stochastic dominance is one of the fundamental concepts to compare random variables (Mann and Whitney (1947); Lehmann (1955)). It introduces a preorder in the space of real random variables. We refer to Muller and Stoyan (2002) for a detailed and comprehensive discussion on stochastic dominance relations. In a stochastic dominance based approach, random variables are compared by a point-wise comparison of some performance functions constructed from their distribution functions. In this study, we utilize the first-order stochastic dominance (FSD) which considers the cumulative distribution function itself as the performance function. Let F_X and F_Y denote the distribution functions of the random variables X and Y, respectively. The FSD relation between X and Y is defined as below:

Definition 2 *A random variable X dominates another random variable Y in the first order; that is, X is stochastically larger than Y, if*

$$F_X(\eta) \leq F_Y(\eta) \quad \textit{for all} \quad \eta \in \mathbb{R}. \tag{3.19}$$

This ordering is denoted by $X \succeq_{(1)} Y$.

It is easy to see that by the definition of the FSD relation we have

$$\Big[X \succeq_{(1)} Y\Big] \Leftrightarrow \Big[\text{VaR}_\alpha(X) \geq \text{VaR}_\alpha(Y) \quad \text{for all } 0 < \alpha \leq 1\Big]. \tag{3.20}$$

We leverage on this fundamental relation between the concepts of VaR and FSD in order to obtain a lower bound on the optimal objective value of VaR-$f(x)$. We consider a finite probability space where the sample space is given by $\Omega = \{\omega_1, \ldots, \omega_{|S|}\}$ with corresponding probabilities $\pi_1, \ldots, \pi_{|N|}$. Let $y_i = Y(\omega_i),\ i \in S$, and $x_i = X(\omega_i),\ i \in S$,

denote the realizations of the random variables Y and X, respectively. In our study, we are interested in the random performance measure of our scheduling problem. In particular, the realizations of the random variable Y are obtained by solving a single-machine problem independently for each scenario. On the other hand, the random variable X denotes the random performance measure associated with the optimal sequence of the problem VaR-$f(x)$. Next, we state formally that $\text{VaR}_\alpha(Y)$ is a lower bound on the optimal VaR obtained by solving VaR-$f(x)$ for any given fixed α.

Proposition 1 *Let Y represent a random variable, where the realization $Y(\omega_i)$ is equal to the objective value associated with the sequence that minimizes a predetermined objective under scenario $i, i \in S$. Furthermore, the random variable X denotes the random performance measure associated with the optimal sequence $\boldsymbol{x}^*$ of the problem VaR-$f(x)$. Then, VaR$_\alpha(Y) \leq$ VaR$_\alpha(X)$ for all $0 < \alpha \leq 1$.*

Proof. $X(\omega_i)$ is the performance measure associated with the sequence $\boldsymbol{x}^*$ under scenario i. Since $\boldsymbol{x}^*$ is a feasible sequence for the problem of minimizing the performance measure under scenario i, we have $X(\omega_i) \geq Y(\omega_i)$ for all $i \in S$. It trivially follows that $P(X \leq \eta) \leq P(Y \leq \eta)$ for all $\eta \in R$, i.e., X dominates Y in the first-order. Consequently, $\text{VaR}_\alpha(Y) \leq \text{VaR}_\alpha(X)$ for all $0 < \alpha \leq 1$ by (3.20). ■

Note that the random variable Y does not have a special interpretation in the context of our problem. It only serves the purpose of obtaining a valid lower bound on the optimal objective function value of our problem.

Calculating the lower bound in Proposition 1 could be performed in $O(|S|n \log n)$ time for the TCT and TWCT objectives by sorting the jobs in Shortest Processing Time (SPT) and Weighted Shortest Processing Time (WSPT) orders, respectively. For total tardiness, a pseudo-polynomial time algorithm by Lawler (1977) could be employed. On the other hand, for TWT this lower bounding scheme is $\mathcal{NP}$-hard since it requires solving $|S|$ instances of the deterministic TWT problem. Although a remedy to this issue would be constructing a lower bound on the optimal TWT under each scenario, we prefered solving the scheduling problems to optimality. This is due to the presence of a very fast algorithm for the single-machine TWT problem proposed by Tanaka et al. (2009)

Finding θ_{UB} is easier than obtaining θ_{LB} since any feasible job sequence could be used to compute an upper bound. To this end, we employ the optimal sequences of the deterministic single scenario problems. For each sequence, the Value-at-Risk associated with the random performance measure is computed and the smallest value over $|S|$ sequences is set as the initial upper bound on θ and $\theta^s, s \in S$.

3.3 Solving the Lagrangian Subproblems

We start our discussion by assuming that $\theta_{UB} = \infty$. We differentiate between three cases in our solution approach for (3.12)-(3.18) depending on the value of the expression $\mu^s + \pi^s$. If $\mu^s < -\pi^s$, then the objective function coefficient of the non-negative variable θ^s is negative in (3.8), and the subproblem is unbounded. Otherwise, if $\mu^s > -\pi^s$ then we can determine the optimal solution by analyzing the dichotomy that results from fixing β^s to zero or one:

$$\beta^s = 1 \rightarrow \quad L^s(\lambda, \mu^s, \boldsymbol{u}) = \sum_{j \in N} \sum_{k \in N} u_{jk} \boldsymbol{H}^s x^s_{jk} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s, \tag{3.21}$$

$$\beta^s = 0 \rightarrow \quad L^s(\lambda, \mu^s, \boldsymbol{u}) = \sum_{j \in N} \sum_{k \in N} u_{jk} \boldsymbol{H}^s x^s_{jk} + (\mu^s + \pi^s)\theta^s + \lambda(\alpha - 1)\pi^s. \tag{3.22}$$

The last two expressions in (3.21) and the final expression in (3.22) are constant terms. Observe that if $(\mu^s + \pi^s) > 0$ and $\beta^s = 1$, then the optimal value of θ^s is $\theta^s_* = \theta_{LB}$, and (3.8) is reduced to (3.21). In this case, the Lagrangian subproblem is an *assignment problem* (AP) which minimizes the first term in (3.21) subject to (3.13), (3.14), and (3.17). The optimal job processing sequence $\boldsymbol{x}_{AP}$ for this case is then obtained by any standard assignment algorithm, such as the famous *Hungarian algorithm*, and the optimal objective value is denoted as

$$D^s(\lambda, \mu^s, \boldsymbol{u}, \beta^s = 1) = z_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s, \tag{3.23}$$

where z_{AP} is the optimal objective value of the assignment problem. Alternatively, if LOF is used the subproblem for $(\mu^s + \pi^s) > 0$ and $\beta^s = 1$ reduces to a *linear ordering problem* (LOP). Unlike the polynomial time AP, LOP is known to be $\mathcal{NP}$-hard (Rafael and Reinelt (2011)). Nevertheless, LOP could be considered as "easy" when compared to directly solving the Lagrangian subproblems.

Unfortunately, if $\beta^s = 0$ then there is a trade-off between the direct cost of the assignment $\boldsymbol{x}^s$ (or linear ordering $\boldsymbol{\delta}^s$) expressed by the first term in (3.22) and the cost $(\mu^s + \pi^s)\theta^s$, where θ^s is set as $\max(\theta_{LB}, f^s(\boldsymbol{x}^s))$ due to the structure of the constraints (3.15), (3.18), and because θ^s appears with a positive coefficient in the objective. Finally, once we relax our initial assumption and set $\theta_{UB} < \infty$, we immediately notice that the feasible region of the Lagrangian subproblems shrink. Therefore, feasibility also becomes an issue. In the next section, we will use these observations in order to compute the opti-

mal solution of subproblems analytically rather than tackling these subproblems using an integer programming solver.

3.3.1 Preprocessing

Depending on the preferred objective function, tackling the subproblems using an integer programming solver may take extreme amounts of time. Therefore, our primary strategy is to avoid solving subproblems as integer programs for some special cases where the optimal objective function value could be easily computed.

Our first observation is regarding the feasibility of the subproblem for the case $\beta^s = 0$. Remember that in Section 3.2, we computed the minimum possible θ^s under each scenario s, call it $f^s_{\min}$. Now that we have $f^s_{\min}$ at hand, we can compare it with the θ_{UB}. If $f^s_{\min} > \theta_{UB}$, then $\beta^s = 0$ cannot be a feasible solution since even the minimum possible $f^s(\boldsymbol{x})$ exceeds the upper bound. Therefore, we can fix $\beta^s = 1$ and solve AP or LOP to get the optimal sequence and the objective function value. Notice that the tighter θ_{UB} is, the more likely that this routine will eliminate subproblems.

A second observation is for θ^s having a zero coefficient in (3.8), i.e. $\pi^s + \mu^s = 0$. In this case, the trade-off described above for $\beta^s = 0$ disappears. Notice that the constant term in (3.22) causes $\beta^s = 0$ to be in the optimal solution unless it is not feasible due to θ_{UB}. The optimal job sequence and the rest of the objective function will be determined again by solving an AP or a LOP.

Finally, we compare the cost of a sequence, $\sum_{j,k \in N} u_{jk} H^s x_{jk}$, for two candidate sequences. First one is the $\boldsymbol{x}_{AP}$ which is the optimal solution of AP. The other candidate $\boldsymbol{x}^s_{\min}$ is the sequence where $f^s(\boldsymbol{x}^s_{\min}) = f^s_{\min}$. If the difference between the cost of the sequence $\boldsymbol{x}_{AP}$ and $\boldsymbol{x}^s_{\min}$ is 0, then we may claim that $\boldsymbol{x}^s_{\min}$ is the optimal solution of the subproblem. Once the sequence is known, it is trivial to compute the other components of the solution, as it was the case before. Similar to above, if LOF is being used, $\boldsymbol{x}_{AP}$ should be substituted with the $\boldsymbol{\delta}_{LOP}$ of the LOP.

Notice that the coefficient of $\boldsymbol{x}$ depends on the scenario index only through $\boldsymbol{H}^s$. Observe that $\boldsymbol{H}^1 = \pi^1 - 1 < 0$ and $\boldsymbol{H}^s = \pi^s > 0$ for $s \geq 2$. Therefore, the optimal job sequence obtained from AP or LOP will be the same in scenarios $s \geq 2$. For these scenarios, we simply minimize $\sum_{j \in N} \sum_{k \in N} u_{jk} x_{jk}$ and then multiply the objective function value with $\boldsymbol{H}^s$. As a result, solving two instances of AP or LOP at a single iteration suffices. In Figure 3.1, the percentage of subproblems that are solved through the described procedures are displayed. The figure is created using the results of 10 representative in-

stances. Here, Case 1, 2 and 3 represent the procedures described above in the order of

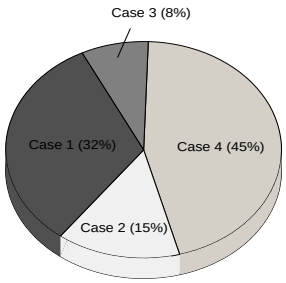


Figure 3.1: Percentages of subproblems that are solved using the proprocessing procedures and by solving mixed integer programs.

explanation. Case 4, on the other hand, represents the subproblems which require mixed integer programs to be solved. It is notable that more than 50% of the subproblems could be solved during the preprocessing stage.

3.3.2 A Polynomially Solvable Case for TCT

If we remove the θ_{LB} and θ_{UB}, then for the TCT objective the subproblems turn out to be polynomial under APDF. Remember that for $\beta^s = 1$ the problem is already polynomial. For $\beta = 0$, we will use the closed form of TCT in (2.4) and plug it into (3.22) which becomes:

$$\begin{aligned} & L^s(\lambda, \mu^s, \boldsymbol{u} \mid \beta^s = 0) \\ & \quad = \sum_{j \in N} \sum_{k \in N} u_{jk} \boldsymbol{H}^s x^s_{jk} + (\mu^s + \pi^s) \left(\sum_{j \in N} \sum_{k \in N} (n-k+1) p_j x^s_{jk} \right) + \lambda(\alpha - 1)\pi^s, \\ & \quad = \sum_{j \in N} \sum_{k \in N} \bar{u}_{jk} x^s_{jk} + \lambda(\alpha - 1)\pi^s, \end{aligned}$$

where

$$\bar{u}_{jk} = u_{jk} \boldsymbol{H}^s + (\mu^s + \pi^s)(n-k+1)p_j.$$

Ignoring the constant part, the problem becomes another AP which is again polynomially solvable. We compare the objective function values of two cases $\beta^s = 1$ and $\beta^s = 0$ where the minimum becomes the optimal solution of the subproblem. Note that we can still put a positive lower bound on θ^s not greater than $\min_{s \in S} f^s_{\min}$, resulting in improved

results. Nevertheless, removing the bounds on θ^s in Lagrangian dual problem resulted in worse outcomes than the lower bound described in Section 3.2. Therefore, we preferred solving mixed integer programs.

3.3.3 Solving Subproblems as Mixed Integer Programs

If the preprocessing procedure cannot solve a subproblem, we have to solve it as a mixed integer program. In this section, we pick the best formulations for TCT, TWCT, and TWT objectives and provide the necessary modifications to (3.13)-(3.18).

VaR-TCT: Our preliminary computational experience suggested that the APDF formulation is superior to LOF when the objective function is TCT. Replacing $f^s(\boldsymbol{x}^s)$ with (2.4) in (3.15), and without the need of additional constraints, we are able to model the VaR-TCT problem.

VaR-TWCT: Due to the job-dependent unit costs w_j, we need additional constraints to express TWCT in APDF. On the other hand, using linear ordering variables one can simply express TWCT as:

$$\sum_{j \in N} w_j C_j = \sum_{j \in N} w_j p_j \sum_{k \in N} \delta_{jk}. \tag{3.24}$$

Once $f^s(\boldsymbol{x}^s)$ in (3.15) is replaced with the expression above, and substituting (3.13)-(3.14) with their counterparts in LOF, the model is complete.

VaR-TWT: Unfortunately, TWT cannot be expressed in closed form using either the assignment or the linear ordering variables. As a result, we require additional constraints to model tardiness as described in Chapter 2. We prefer using LOF since our preliminary studies suggested that the computational performance is better when compared to the performance of subproblems formulated with APDF. For completeness, we present the Lagrangian subproblem formulated using LOF below:

$$\begin{aligned}
& \min \quad L^s(\lambda, \mu^s, \boldsymbol{u}) && && && (3.25)\\
\text{subject to} \quad & \delta^s_{jj} = 1, && && \forall j \in N, && (3.26)\\
& \delta^s_{jk} + \delta^s_{kj} = 1, && && 1 \le j < k \le n, && (3.27)\\
& \delta^s_{jk} + \delta^s_{kl} + \delta^s_{lj} \le 2, && \forall j,\ k,\ l \in N\ : && j \ne k,\ k \ne l,\ l \ne j, && (3.28)\\
& C^s_j = \sum_{k \in N} p^s_k \delta^s_{kj}, && && \forall j \in N, && (3.29)\\
& T^s_j \ge C^s_j - d_j, && && \forall j \in N, && (3.30)
\end{aligned}$$

$$T_j^s \geq 0, \qquad \forall j \in N, \quad (3.31)$$

$$\sum_{j \in N} w_j T_j^s - \theta^s \leq T_{\max}^s \beta^s, \quad (3.32)$$

$$\beta^s \in \{0,1\}, \quad (3.33)$$

$$\delta_{jk}^s \in \{0,1\}, \qquad \forall j,k \in N. \quad (3.34)$$

$$\theta_{LB} \leq \theta^s \leq \theta_{UB}. \quad (3.35)$$

We provide the pseudocode of our proposed method for solving the Lagrangian subproblems in Algorithm 1.

3.3.4 Parallel Programming

In the stochastic programming literature, parallelization of stochastic optimization methods receives considerable attention due to the independent structure of the subproblems. Ruszczyński (1993) uses the notion of parallel computing in order to solve a multi-stage stochastic inventory management problem. Birge et al. (1996) similarly focus on multi-stage problems where the decomposed components of the scenario tree are tackled using independent processors. Further, Linderoth and Wright (2003) work on algorithms for two-stage stochastic linear programming models with recourse on a grid computing platform.

Similar to such studies, our single-stage stochastic programming model could benefit from the parallel computing of subproblems. Noting that the largest portion of time in our solution algorithm is spent on solving subproblems, parallel programming offers a great potential on improving the overall performance. In order to incorporate parallelization, at every iteration we gather all the subproblems, which could not be solved analytically, into a set. At every step, we pick K subproblems from this set and solve them using K-many processors. Once this batch of subproblems are all completed, we pick K more subproblems from the remaining of the set and continue until all the subproblems are solved. This allows us to solve the subproblems in $\lceil \frac{|S|-A}{K} \rceil$ steps where A is the number of subproblems that could be solved analytically. Compared to the serial algorithm, in which we require $|S| - A$ steps to solve the integer programs at every iteration, parallel computing offers a good advantage as the number of processors grows.

An important point should be made regarding the balancing of workload among the processors. Our subproblems do not necessarily have similar solution times. In fact, the parameters of a scenario could make a subproblem relatively more difficult compared to

Algorithm 1: Solving the Lagrangian subproblems.

input : Values of the dual variables λ, $\boldsymbol{\mu}$, and $\boldsymbol{u}$.

output: The optimal objective value of $D^s(\lambda, \mu^s, \boldsymbol{u})$ and the optimal solution $\boldsymbol{x}^s_*$, β^s_*, θ^s_* for all scenarios $s \in S$.

Solve two assignment problems, retrieve $\boldsymbol{x}_{AP^+}, \boldsymbol{x}_{AP^-}$ and z_{AP^+}, z_{AP^-};

/* $\boldsymbol{x}_{AP^-}$ and z_{AP^-} will be used in scenario 1, and their positive counterparts will be used for the other scenarios. For brevity, in the description below we use $\boldsymbol{x}_{AP}$ for both $\boldsymbol{x}_{AP^+}$ and $\boldsymbol{x}_{AP^-}$ and z_{AP} for both z_{AP^+} and z_{AP^-}. */

for $s = 1$ **to** $|S|$ **do**

  if $f^s_{\min} \leq \theta_{UB}$ **then**

    if $\pi^s + \mu^s = 0$, *and* $f^s(\boldsymbol{x}_{AP}) \leq \theta_{UB}$ **then**

      $\beta^s = 0; \quad \theta^s = \max\{\theta_{LB}, f^s(x_{AP})\}; \quad \boldsymbol{x}^s = \boldsymbol{x}_{AP}$;

      continue with the next scenario;

    end

    if $\boldsymbol{u}^\top \boldsymbol{x}_{AP} \boldsymbol{H}^s - \boldsymbol{u}^\top \boldsymbol{x}^s_{\min} \boldsymbol{H}^s = 0$ **then**

      /* $x^s_{\min}$ is an alternate optimal solution to the assignment problem. */

      if $D^s(\lambda, \mu^s, \boldsymbol{u} \mid \beta^s = 0) < D^s(\lambda, \mu^s, \boldsymbol{u} \mid \beta^s = 1)$ **then**

        $\beta^s = 0; \quad \theta^s = \max\{\theta_{LB}, f^s(\boldsymbol{x}^s_{\min})\}; \quad \boldsymbol{x}^s = \boldsymbol{x}^s_{\min}$;

      else

        $\beta^s = 1; \quad \theta^s = \theta_{LB}; \quad \boldsymbol{x}^s = \boldsymbol{x}_{\min}$;

      end

      continue with the next scenario;

    end

    /* If the subproblem is not solved up to this point, then we have to solve an integer program. */

    Solve integer program;

  else

    $\beta^s = 1; \quad \theta^s = \theta_{LB}; \quad \boldsymbol{x}^s = \boldsymbol{x}_{AP}$;

  end

end

the other subproblems. In order to efficiently utilize the processors, one must carefully balance the workload and avoid assigning difficult subproblems to the same processor. In our study, before solving the integer programs, we sorted the subproblems by looking at their most recent solution times. We used the Longest Processing Time (LPT) rule which is a pretty good approximation to minimizing the makespan on parallel processors (see Pinedo (1995)). This strategy not only reduces the makespan, but also decreases the probability of leaving a processor idle by assigning similar difficulty subproblems to different processors. As a result, the workload is more balanced and we are able to utilize the given processors more effectively.

3.4 Solving the Lagrangian Dual Problem

In order to attain the best lower bound on VaR-$f(x)$, several methods are proposed in the literature. Among all, the simplest is called the *subgradient method*. In this method, at every iteration the Lagrangian subproblems are solved. Then, the dual variables are updated in the opposite direction of the subgradient using an appropriate step size. More information on the algorithm and the step size rules for minimization can be obtained from Wolsey (1998). In addition, several sophisticated algorithms have been developed, such as the *bundle methods* (see Hiriart-Urruty and Lemaréchal (1993)). In our preliminary studies, we have tried both of these methods in order to solve our Lagrangian dual problem. However, we have faced several convergence issues which prevented the algorithm to reach to a solution in a sufficient amount of time. Therefore, we implemented another strategy known as the *cut-generation algorithm*. This algorithm is based on the idea that the Lagrangian dual problem (3.11) can be equivalently represented by a linear program:

$$\max_{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}} \quad D(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \quad \max_{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}} \sum_{s \in S} \eta^s \tag{3.36}$$

subject to

$$\eta^s \leq L^s(\lambda, \mu^s, \boldsymbol{u} \mid \boldsymbol{x}^s, \beta^s, \theta^s) \qquad \begin{array}{r} \forall s \in S, \\ \forall (\boldsymbol{x}^s, \beta^s, \theta^s) \in \boldsymbol{\Phi}^s \end{array} \tag{3.37}$$

$$\lambda \geq 0, \tag{3.38}$$

$$\sum_{s \in S} \eta^s \leq \theta_{UB} \tag{3.39}$$

$$\mu^s \geq -\pi^s \qquad \forall s \in S, \tag{3.40}$$

$$\sum_{s \in S} \mu^s = 0. \tag{3.41}$$

This linear program (3.36)-(3.41) is called the *master problem*. The right hand side of the term in (3.37) represents the Lagrangian function described in (3.8) evaluated at the solution $(\boldsymbol{x}^s, \beta^s, \theta^s)$ under scenario s. Here, $\boldsymbol{\Phi}^s$ represents the set of all feasible solutions under scenario s. Fortunately, we do not need to generate all elements of the set $\boldsymbol{\Phi}^s$ $\forall s \in S$, but a small portion of it will suffice to obtain the optimal objective function value of the master problem. As a matter of fact, instead of solving the master problem, we solve a *restricted master problem* where we start with a small subset of the constraints (3.37), also known as 'cuts'. The algorithm works iteratively where the Lagrangian subproblems (3.10) are solved at each iteration, a set of new cuts is constructed based on this solution and appended to the restricted master problem, then the duals are updated according to the new solution of the restricted master problem. The constraints (3.39) ensure that the restricted master problem is always bounded, where θ_{UB} is an upper bound on the VaR and the optimal objective function value of the master problem.

As the $\boldsymbol{\Phi}^s$ is not completely generated, the master problem may contain extreme rays. Therefore, we put an upper bound (3.39) on the objective function of the master problem to avoid unboundedness. In order to increase the stability of the algorithm, we eliminated unbounded subproblems (see Section 3.3) using the constraint (3.40). In order to preserve the relation described in (3.6), constraint (3.41) is enforced. We also imposed (3.38) since the dualized constraint is in inequality form. Finally, if LOF is used instead of APDF, due to (3.26), the following set of constraints must be appended to the master problem:

$$u_{jj} = 1 \quad \forall\, j \in N.$$

We note that (3.36)-(3.41) is a "multi-cut" formulation. An alternative would be using a "single-cut" formulation where the constraint (3.37) should be replaced by

$$\eta \leq \sum_{s \in S} L^s(\lambda, \mu^s, \boldsymbol{u} \mid \boldsymbol{x}^s, \beta^s, \theta^s) \qquad \big((\boldsymbol{x}^1, \beta^1, \theta^1), ..., (\boldsymbol{x}^{|S|}, \beta^{|S|}, \theta^{|S|})\big) \in \boldsymbol{\Phi},$$

where

$$\boldsymbol{\Phi} = \boldsymbol{\Phi}^1 \times \boldsymbol{\Phi}^2 \times ... \times \boldsymbol{\Phi}^{|S|}.$$

Such a modelling will result into less number of constraints in the master problem, making it easier to solve. On the other hand, this increases the required number of iterations for convergence tremendously. In our case, solving the subproblems is more expensive

than solving a relatively difficult linear program. Therefore, our primary objective is to decrease the number of iterations. In consequence adding $|S|$-many cuts at every iteration is more favorable to our cause.

3.4.1 Updating Bounds & Cuts

Recall that the cut generation algorithm creates feasible job processing sequences through its progress. At every iteration, we use the sequences from the solutions of Lagrangian subproblems, and compute the VaR associated with these sequences. We compare these VaR measures with the best primal solution that we have obtained so far. If one of these sequences produces a better objective function value, we update our best primal solution. Similarly, one can update the best lower bound attained by using the objective function value of the Lagrangian problem. Note that a better primal solution and a better lower bound could be used to update the θ_{UB} and θ_{LB} in the master problem and the Lagrangian subproblems. In fact, when the bounds on $\theta^s \; \forall s \in S$ are updated, we observe a considerable improvement in the convergence rate of the algorithm and the quality of the terminal lower bound. This observation is supported by the fact that tighter bounds reduce the size of the convex hull of Lagrangian subproblems. In consequence, the objective function value associated with a sequence in the modified Lagrangian subproblem will be no smaller than the objective function value in the original subproblem. Therefore, the optimal objective function value of the modified master problem will be at least as large as the optimal objective function value of the current master problem.

One major problem that arises due to this update is the infeasibility regarding the previously appended cuts. More specifically, a value of θ^s obtained at a previous iteration may not necessarily be feasible due to the new θ_{LB} and θ_{UB}. However, this θ^s is already appended as a cut to the restricted master problem. As a result, we over-constraint the master problem, so the algorithm may terminate prematurely. In order to fix this issue, we have to ensure that the previously appended cuts are feasible with respect to the new boundaries. Notice that we only need to check the values of θ^s's. If they are turn out to be infeasible, we either have to delete them from the master problem or reoptimize the solution. In our study, we prefer to extract the sequence from the cut, which contains infeasibility, then solve the Lagrangian subproblem without changing this sequence. In other words, we recompute the value of β^s and θ^s for the previously generated x^s. We empirically observed that using the initial values of the dual variables, $\lambda = 0$, $\boldsymbol{\mu} = \boldsymbol{u} = \mathbf{0}$, while reoptimizing the β^s and θ^s provided the best results in terms of convergence speed.

If this cut-update routine is performed whenever one of the bounds is updated, we ensure that the algorithm converges to the true optimal solution of the Lagrangian dual problem. Furthermore, the routine allowed us to update θ_{UB} and θ_{LB} during an intermediate iteration, hence considerably improved our results.

3.4.2 Updating the Non-anticipativities

Remember that in Section 3.1 we have used the set of non-anticipativity constraints in (3.2) for $\theta^s, \forall s \in S, s \neq 1$. Obviously, the choice of using θ^1 in the left hand side of this constraint is arbitrary. In fact, any θ^s could be used instead of θ^1. However, it turns out that the optimal objective function value of the Lagrangian dual problem is highly dependent on the scenario that is used in the left hand side of these constraints. This is equivalent to saying that the quality of our lower bound depends on the non-anticipativity that we pick. Notice that this only applies to the relaxed version of the problem. In other words, the choice of non-anticipativity cannot affect the optimal objective function value of the non-relaxed problem. We set up a computational study in order to find a scenario k for the non-anticipativity constraints redefined below:

$$\theta^k = \theta^s \qquad\qquad \forall s \in S, s \neq k.$$

We have identified that the scenario, which defines the VaR in the initial lower bound obtained by optimally solving the underlying deterministic problems (see Section 3.2), should be selected as scenario k. In fact, in almost all instances this selection resulted in the best lower bounds that we have ever achieved for those instances. Consequently, as an initialization step, we incorporate this update on the non-anticipativity constraints to our algorithm so that the terminal quality of our results is improved. Notice that the implementation could easily be handled by a simple re-indexing of the scenarios. Once the objective function values of the deterministic problems are obtained, it is sufficient to swap the scenario k with scenario 1 in the data just before initializing subproblems.

3.4.3 Optimal Solution of the Lagrangian Dual Problem for a Special Case of the Lagrangian Function

In this section, we present a proof regarding the optimal solution of the Lagrangian dual problem when the direct cost of assignment, $\sum_{j \in N} \sum_{k \in N} u_{jk} H^s x_{jk}$, is neglected (i.e. when $\mathbf{u} = \mathbf{0}$).

Proposition 2 *If the dual variables $\boldsymbol{u}$ are restricted to zero, then the lower bound on VaR obtained from the Lagrangian Dual problem* (3.11) *is no better than θ^0_{LB} where θ^0_{LB} is obtained by optimally solving the deterministic problems for each scenario and computing the $(1-\alpha)$-quantile of the objective function values as described in Section 3.2. That is,* $\max_{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}=\mathbf{0}} D(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \theta^0_{LB}$.

Proof. We claim that the optimal solution to $\max_{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}=\mathbf{0}} D(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is given by $\lambda^* = 0,\ \boldsymbol{\mu}^* = (1-\pi^1, -\pi^2, ..., -\pi^{|S|}),\ \boldsymbol{u}^* = 0$. It is a well known fact that $D(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is a non-differentiable piecewise linear and concave function. Therefore, we can complete the proof by showing that the zero vector is a subgradient of $D(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ at $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$. Our strategy is to first show that $D(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*) = \theta^0_{LB}$ and then prove that $\mathbf{0}$ is a subgradient at $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$. Define $S^0 = \{s \mid f^s(x^s_{\min}) \leq \theta^0_{LB}\}$, where $f^s(x^s_{\min})$ and $x^s_{\min}$ are the optimal objective function value and the optimal solution of the corresponding deterministic problem for scenario s. Similarly, define $S^1 = \{s \mid f^s(x^s_{\min}) > \theta^0_{LB}\}$, so that $S = S^0 \cup S^1$. Further, assume that the scenarios are re-indexed according to Section 3.4.2, so that scenario 1 has an objective function value equal to θ^0_{LB}.

Observe that the objective functions of the Lagrangian subproblems reduce to

$$L^s(\lambda, \mu^s, \boldsymbol{u}) = \begin{cases} \theta^1 & \text{for } s = 1 \\ 0 & \text{for } s \geq 2. \end{cases}$$

at $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$. Thus, we have $D(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*) = \sum_{s \in S} D^s(\lambda^*, \mu^s_*, \boldsymbol{u}^*) = \theta^1_*$, where $\theta^1_* = f^1(x^1_*) = \theta^0_{LB}$ and $\beta^1_* = 0$. For other scenarios, we specify the optimal subproblem solutions as

$$\begin{cases} \theta^s_* = \theta^0_{LB},\ \beta^s_* = 0 & \text{if } s \in S^0 \setminus \{1\} \\ \theta^s_* = \theta^0_{LB},\ \beta^s_* = 1 & \text{if } s \in S^1 \end{cases}.$$

For these subproblem solutions, $\mathbf{d^1} = (d^1_\lambda, \mathbf{d^1_\mu}, \mathbf{d^1_u})$ is a subgradient at $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$, where

$$\begin{aligned} d^1_\lambda &= \sum_{s \in S} \pi^s \beta^s_* - (1-\alpha) = \sum_{s \in S^1} \pi^s - (1-\alpha) \leq 0 \\ \mathbf{d^1_\mu} &= (\theta^2_* - \theta^1_*,\ \theta^3_* - \theta^1_*,\ ...,\ \theta^{|S|}_* - \theta^1_*) = \mathbf{0} \\ \mathbf{d^1_u} &= \mathbf{0}. \end{aligned}$$

A crucial observation is that the subproblems have many alternate optimal solutions. In particular, $\theta^s_* = \theta^0_{LB}$, $\beta^s_* = 1\ \forall s \in S$ is also optimal for the Lagrangian subproblems at

$(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$. We compute a second subgradient $\mathbf{d^2} = (d^2_\lambda, \mathbf{d^2_\mu}, \mathbf{d^2_u})$ at $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$ with:

$$
\begin{aligned}
d^2_\lambda &= \sum_{s \in S} \pi^s \beta^s_* - (1 - \alpha) = \alpha > 0 \\
\mathbf{d^2_\mu} &= \mathbf{0} \\
\mathbf{d^2_u} &= \mathbf{0}.
\end{aligned}
$$

Since the convex combination of two subgradients is another subgradient, we are ensured that $\mathbf{d^3} = \mathbf{0}$ is a subgradient at $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$. Clearly, we can always identify $m_1, m_2 \geq 0, m_1 + m_2 = 1$ such that $m_1 d^1_\lambda + m_2 d^2_\lambda = m_1 \left(\sum_{s \in S^1} \pi^s - (1 - \alpha) \right) + m_2 \alpha = 0$. ■

3.4.4 Dual Stabilization

Although theoretically correct, the straightforward implementation of the cut-generation algorithm may lead to instability in terms of convergence. More specifically, the objective function value of the Lagrangian dual at the current iteration might be significantly better than the objective function value in the next iteration (Kallehauge et al. (2006)). As a matter of fact, even initializing the cut-generation algorithm with the 'best' values of the dual variables has little effect on convergence (Frangioni and Gendron (2010)). This is due to the incapability of the Lagrangian problem to generate the necessary subset of $(\boldsymbol{x}^s, \beta^s, \theta^s)$ to prove the optimality of the Lagrangian dual problem. Therefore, a remedy to this issue would be forcing the algorithm to explore the region where improvement on the objective function value of Lagrangian dual problem is more likely to be observed.

In our study, we have also observed high fluctuations in the values of the dual variables and consequently in the objective function value of the Lagrangian problem at every iteration. Due to such instability, the algorithm requires a large number of iterations to be able to converge to a solution. In order to prevent the instability, several stabilization functions are proposed in the literature. One of these approaches, the *Box-Step Method* or the *Trust Region Method* (Frangioni and Gendron, 2010; Kallehauge et al., 2006), confines the dual variables into a 'box'. This method prevents the duals from taking values far from the center of the box, known as the *stability center*. Another type of stabilization functions is the *linear penalty functions* (Frangioni and Gendron (2010)) where the dual variables are penalized according to their distance from the stability center. In fact, the imposing linear penalty functions could be considered as an extension of the box-step method where the cost is ∞ across the boundaries of the box. In both methods, we update the stability center whenever a sufficient improvement in the best objective function value

of the Lagrangian dual is observed *(Serious Step)*. If there is no sufficient improvement, the center is kept the same *(Null Step)* to explore the current region more. A more elaborate discussion on the stabilization functions and additional methods could be found in Frangioni and Gendron (2010). In our study, we adapt the stabilization logic described in Kallehauge et al. (2006) with additional linear penalty functions on the dual variables. In particular, we use the Box-Step Method and linear penalty functions together.

We have empirically observed that the major source of instability is due to $\boldsymbol{u}$, therefore employed dual stabilization techniques only on $\boldsymbol{u}$. In addition, we have also observed that many components of $\boldsymbol{u}$ are 0 in the optimal solution of the Lagrangian dual problem. Therefore, we fixed the stability center of every u_{jk} to 0 which not only improved the stability of the algorithm but also the performance of the subproblems. This is because the closer u_{jk}'s are to 0, the more our subproblem looks like the underlying deterministic problem. Below we present the stabilization function $\Psi(\boldsymbol{u})$ we used in our study:

$$\Psi(\boldsymbol{u}) = \sum_{j \in N} \sum_{k \in N} \Psi_{jk}(u_{jk}),$$

where

$$\Psi_{jk}(u_{jk}) = \begin{cases} +\infty, & \text{if} \quad u_{jk} > \Delta^+ \\ -u_{jk}\Gamma, & \text{if} \quad 0 \leq u_{jk} \leq \Delta^+ \\ u_{jk}\Gamma, & \text{if} \quad \Delta^- \leq u_{jk} \leq 0 \\ -\infty & \text{if} \quad u_{jk} < \Delta^- \end{cases}.$$

Here Δ represents the width of the box that our dual variables are restricted into, and Γ is the linear penalty cost term. We note that the stabilization function $\Psi(\boldsymbol{u})$ is directly added to the objective function of the restricted master problem. This function is a *4-piecewise linear penalty function* which can be modeled by defining two copies of the dual variables u_{jk}, namely u^+_{jk} and u^-_{jk}, $\forall j, k \in N$. We can then express $|u_{jk}|$ as $u^+_{jk} + u^-_{jk}$, and u_{jk} as $u^+_{jk} - u^-_{jk}$. Notice that the number of variables in the master problem only increases by $|N|$ which does not have any significant effect on the solution times. We illustrate the our stabilizing function in Figure 3.2.

We will now describe our stabilization scheme. We propose a 3-phase stabilization strategy. At the start of the algorithm, we set $\boldsymbol{u} = \mathbf{0}$ (Phase I). In Section 3.4.3, we have already given the optimal solution of this case. Therefore, we only need to solve the master problem to get the new values of λ and $\boldsymbol{\mu}$. Once the gap between the objective function value of master problem, call it z_{LD} and θ_{LB} is closed, we set the new boundaries for $\boldsymbol{u}$ such that $\Delta^- \leq \boldsymbol{u} \leq \Delta^+$ (Phase II). From this point on, any u_{jk} could take nonzero

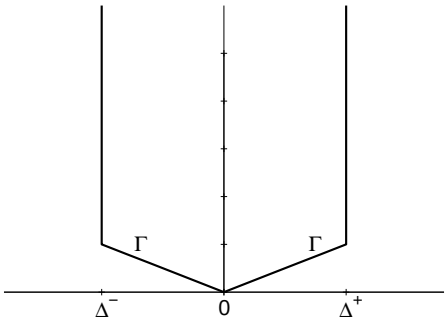


Figure 3.2: Stabilizing function on $\boldsymbol{u}$.

values. Therefore, the stabilization effect of the linear penalty functions is observed from this point on.

At every iteration, we compute a parameter ρ which is defined as:

$$\rho = \frac{D^k - D^{k-1}}{z_{LD}^{k-1} - D^{k-1}}.$$

Here, D^k and D^{k-1} represent the value of the dual function at iteration k and $k-1$, respectively, and z_{LD}^k is the objective function value of the master problem at iteration k. Observing $\rho = 1$ is a strong indication that the current bounds on $\boldsymbol{u}$ are restricting the potential improvement of D, therefore we increase the width of the box, Δ. On the other hand, if $\rho < 0$, then we decrease Δ to explore the region more. Whenever the bounds are updated, we increase the value of the linear penalty costs in the new problem to keep the u_{jk} close to 0. Finally, after solving the master problem, if $\frac{z_{LD}-D}{D}$ is within a predetermined gap, we reduce the linear penalty costs to prevent the algorithm from terminating prematurely. If these costs become smaller than 10^{-4}, we completely remove them.

In order to move on to Phase III, one of two termination conditions of Phase II must be fulfilled. The first condition occurs when the optimality gap, $\frac{z_{LD}-D}{D}$, reaches a very tight tolerance value. In our study, we set this value to 10^{-8}. This suggests that in Phase II, we have to solve the Lagrangian dual problem very close to optimality. This phase is essential for the achievement of large improvements on the Lagrangian dual problem since such improvements could only be achieved by making small improvements at the initial stages. Note that the tolerance value suggested in this phase is too strict, therefore may not be achieved within a reasonable amount of iterations. Our second condition is a remedy to this problem. This condition is related to the number of times the bound

attained in Phase I is exceeded in Phase II. First, we wait for the algorithm to exceed the lower bound of Phase I for at least 15 times. We expect the larger improvements on the objective function value of the Lagrangian dual problem during and after this stage. In order to terminate, we make sure that no sufficient improvements will be observed in this phase any more. After the lower bound of Phase I is exceeded 15 times, we wait for another 15 iterations. Once these iterations are also carried out, at each iteration we check whether the improvement within 15 iterations is larger than a limit, $\%0.1$. If the improvement is less than this limit, we move on the Phase III.

As described previously, one of the conditions to enter Phase III is achieved when the optimality gap is below a strict tolerance. In such a case, we immediately remove the bounds on $\boldsymbol{u}$ to provide room for improvement for the objective function value of the Lagrangian dual problem. On the other hand, if Phase II ends due to our second condition we keep these bounds until termination. Furthermore, we update the box width according to the value of ρ as described previously. We continue in Phase III until the linear penalty functions on $\boldsymbol{u}$ are removed and a relatively loose optimality gap (10^{-3}) has been achieved. This is the final termination criteria that we check before terminating the whole algorithm.

3.4.5 Suboptimal Cuts

An important note is that the cut-generation algorithm can also utilize any suboptimal solutions for a given values of the dual variables. In other words, we can use any feasible job sequence, compute the corresponding θ^s and β^s, then append the solution as a new cut to the master problem. Using such suboptimal cuts is equivalent to multiple-pricing in column generation where a set of non-basic variables are selected instead of a single non-basic variable (see Chvátal (1983)). This strategy is fruitful when it comes to decreasing the number of iterations that our algorithm requires for convergence. In order to implement it, we gather all the suboptimal solutions from the solution pool of the mixed integer program solver we use at every iteration, and append them to the master problem. In order to control the amount of suboptimal cuts, we impose a limit on the number of suboptimal solutions, and their gap of optimality. In our study, we preferred at most 10 suboptimal solutions per scenario which should be within at most 40% optimality gap.

A further use of suboptimal cuts would be when the integer programming subproblem takes too much time to be solved to optimality. Occasionally, the objective function of a subproblem can make it extremely difficult. In fact, a single subproblem could consume

more than three or four times larger amount of seconds than the total solution times of all other subproblems. On the other hand, we could avoid solving such subproblems to optimality by utilizing a suboptimal solution of the Lagrangian subproblem. Neame et al. (2000) present an outer approximate subdifferential method applied to an uncapacitated facility location problem where they use a dynamically updated approximation parameter. In our study, we impose a time limit on our integer programming subproblems resulting in suboptimal solutions with uncertain optimality gaps. In order to increase the quality of the suboptimal solutions, we can use of the optimal solution of the subproblem for $\beta^s = 1$ by a simple comparison of its objective function value and the value returned from the prematurely terminated integer program. Notice that the time limit may result into poor convergence. Therefore, we expand the limit by 50% at every iteration if the subproblem could not be solved to optimality. We also note that there will be at least one feasible solution to the integer programming subproblems, coming from previous iterations, which prevents the program from terminating with no solution at hand.

3.4.6 Cut Management

Since we are adding at least $|S|$ constraints to our master problem at every iteration, it is likely that this linear program will slow down the cut-generation algorithm as the number of iterations grow. Further, the linear programming solver faces numerical difficulties due to the huge number of constraints. In order to prevent these, we developed a cut management strategy. Once the master problem is solved, we analyze values of the dual variables which correspond to the cuts in the master problem. Note that, observing a dual variable, which has a value of 0, suggests that the corresponding constraints is likely to be inactive. We record a statistic for each cut in the restricted master problem which counts the successive number of iterations that the corresponding dual variable is fixed at 0. Once this statistic exceeds 5 iterations, we labeled the cut as redundant. However, we do not carry out an immediate deletion since removing cuts at every iteration may slow down the convergence. Instead, we remove constraints from the master problem every 5 iterations. To summarize, at every 5 iterations, if we observe a constraint with an associated dual fixded at 0 for more than 5 iterations, we remove that constraint. Otherwise, we keep it. This approach successfully prevents the growth of the number of constraints, resulting in steady and fast solution times for the master problem.

3.4.7 The Cut-Generation Algorithm

Below, we provide the pseudocode of our algorithm for solving the Lagrangian dual problem.

Algorithm 2: Solving the Cut-Generation Algorithm.

```
1  Compute initial θ_LB and θ_UB;                          // Section 3.2
2  Update non-anticipativities;                            // Section 3.4.2
3  while termination criteria are not satisfied do
4      Compute Σ_{s∈S} D(λ, μ, u);                         // Algorithm 1
5      Update θ_LB and θ_UB;
6      Add optimality cuts;
7      Add suboptimal cuts;
8      if bounds are updated then
9          Update cuts;                                    // Section 3.4.1
10         Update subproblems and master problem;
11         Increase Γ;                                     // Section 3.4.4
12     end
13     Adjust Δ;                                           // Section 3.4.4
14     Solve the restricted master problem;
15     Eliminate redundant cuts;
16 end
```

Chapter 4

Computational Study

The goals of our computational study are two-fold. In the first part, we demonstrate that the cut-generation algorithm described in Chapter 3 produces good lower bounds for the VaR measure on the three random performance measures, TCT, TWCT and TWT. Furthermore, the results indicate that the algorithm yields feasible solutions of very high quality for the risk-averse single-machine scheduling problems for almost all instances that we have experimented. In the second part, the value of the proposed risk-averse model is investigated with respect to that of a risk-neutral model.

All runs were conducted on a machine with Intel®Core™i7 960 3.20GHz CPU and 24 GB of memory. The mathematical programming formulations were solved by `CPLEX 12.4`, and the cut-generation algorithm was implemented in C++. Further, the Boost Library was used to implement multi-threading in the cut-generation algorithm. In this study we allowed only two cores to be utilized simultaneously. Note that, `CPLEX 12.4` is able to utilize more than one core to solve a single mathematical program. As our parallelization strategy aims to solve multiple subproblems at the same time, we limited `CPLEX` to use only a single core while solving the mixed integer program subproblems. However, when solving the master problem, assignment or linear ordering problems, we allowed `CPLEX` to utilize up to two cores.

4.1 Generation of problem instances

While our modeling framework allows for randomness in all problem parameters, we focus on the uncertainty in the processing times in our computational study as justified by the discussion in Chapter 1. For each instance, we generate a set of equally likely

scenarios representing the joint realizations of the processing times by adding negative or positive perturbations to each estimated processing time $\hat{p}_j$, where $\hat{p}_j$ follows an integer uniform distribution $U[1, 100]$ for $j = 1, \ldots, n$. To this end, let ε_j denote the random perturbation for job j, where ε_j^s is the realization of ε_j for scenario s. Then, the processing time of job j under scenario s is given by $p_j^s = \hat{p}_j + \varepsilon_j^s$. In our first set of experiments, we set $\varepsilon_j \sim U(-\hat{p}_j/4, \hat{p}_j/3)$, which results in $E(\hat{p}_j + \varepsilon_j) = \hat{p}_j + \hat{p}_j/24$ and a coefficient of variation (CV) of 0.16. CV is a normalized measure of dispersion and is defined as $CV(\hat{p}_j + \varepsilon_j) = standard\ deviation(\hat{p}_j + \varepsilon_j)/E(\hat{p}_j + \varepsilon_j)$ for the processing time of job j. We also generated an additional data with a higher CV (0.26) to further analyze the value of our risk-averse model in Section 4.3. This data was generated by drawing ϵ_j from $U(-\hat{p}_j/4, \hat{p}_j)$.

In the literature, it is well established that the tightness and the range of the due dates is a primary determinant of difficulty for due date related problems. Thus, by following the popular scheme of Potts and van Wassenhove (1982), we first generate the due dates from a discrete uniform distribution $[\lceil(1 - TF - RDD/2) \times \bar{P}\rceil, \lceil(1 - TF + RDD/2) \times \bar{P}\rceil]$, where $\bar{P}$ is the sum of the expected processing times, i.e., $\bar{P} = \sum_{j=1}^{n} \sum_{s \in S} \pi^s p_j^s$. The tardiness factor TF is a rough estimate of the proportion of jobs that might be expected to be tardy in an arbitrary sequence (Srinivasan (1971)) and is set to 0.4 and 0.6. Hard instances generally result from small values of TF (see Bulbul et al. (2007); Sen (2010)). The due date range factor RDD is set to 0.8 to have mediocre contention around the mean due date. The weights are drawn from an integer uniform distribution $U(10, 20)$.

4.2 Computational Performance of the Cut-Generation Algorithm

In the first part of our study, we generate 5 instances for each combination of TF$= 0.4, 0.6$, $n = 10, 15, 20, 30$, and $|S| = 50, 100, 150, 200$, as described in the previous section. The risk parameter $\alpha = 0.90$. For each instance, we run our cut-generation algorithm and use CPLEX to solve the VaR-$f(x)$ problem to optimality. The results averaged over 5 instances appear in Tables 4.1-4.3 for TCT, TWCT and TWT performance measures, respectively.

The time limit for CPLEX is set to 3600 seconds, and if optimality is not proven in the time allotted, then we record both the best lower bound and the incumbent solution available. Similarly, we impose a time limit of 3600 seconds on the cut-generation algorithm,

		$\|S\|$			
		50	100	150	200
$\mathbf{n = 10}$	**UB Gap**	0.2%	0.0%	0.0%	0.0%
	LB Gap	-1.0%	-1.1%	-1.2%	-1.5%
	Time (Cut-Gen)	5.9	11.5	31.4	17.9
	Time (CPLEX)	0.1	0.2	0.5	0.6
$\mathbf{n = 20}$	**UB Gap**	0.7%	0.5%	0.6%	0.4%
	LB Gap	-1.6%	-1.3%	-1.5%	-1.5%
	Time (Cut-Gen)	19.2	39.8	34.5	60.7
	Time (CPLEX)	1.4	2.1	6.5	6.3
$\mathbf{n = 30}$	**UB Gap**	0.9%	0.9%	0.9%	0.7%
	LB Gap	-1.4%	-1.4%	-1.7%	-1.4%
	Time (Cut-Gen)	43.0	71.5	120.7	180.2
	Time (CPLEX)	19.0	22.9	102.8	104.9

Table 4.1: Effectiveness of the cut-generation algorithm under TCT performance measure ($\alpha = 0.90$).

		$\|S\|$			
		50	100	150	200
$\mathbf{n = 10}$	**UB Gap**	0.1%	0.1%	0.0%	0.0%
	LB Gap	-1.0%	-1.1%	-1.4%	-1.2%
	Time (Cut-Gen)	4.8	10.2	13.5	15.1
	Time (CPLEX)	0.1	0.1	0.2	0.4
$\mathbf{n = 20}$	**UB Gap**	0.5%	0.4%	0.3%	0.2%
	LB Gap	-1.1%	-1.0%	-1.4%	-1.6%
	Time (Cut-Gen)	47.7	72.9	115.5	152.7
	Time (CPLEX)	0.8	1.1	3.3	7.1
$\mathbf{n = 30}$	**UB Gap**	0.6%	0.5%	0.4%	0.3%
	LB Gap	-1.0%	-1.1%	-1.1%	-1.2%
	Time (Cut-Gen)	311.3	570.2	1381.5	1356.2
	Time (CPLEX)	11.5	18.1	95.6	171.9

Table 4.2: Effectiveness of the cut-generation algorithm under TWCT performance measure ($\alpha = 0.90$).

			$\|S\|$			
			50	100	150	200
TF = 0.4	$\mathbf{n = 10}$	**UB Gap**	0.0%	0.0%	0.0%	0.0%
		LB Gap	-2.6%	-0.5%	-1.7%	-0.9%
		Time (Cut-Gen)	21.7	9.1	75.6	33.4
		Time (CPLEX)	3.4	34.7	69.1	120.4
	$\mathbf{n = 15}$	**UB Gap**	0.0%	1.6%	1.9%	1.8%
		LB Gap	-1.1%	-1.6%	-2.1%	-1.8%
		Time (Cut-Gen)	1199.6	1009.1	1183.9	2134.6
		Time (CPLEX)	207.7	1926.5	2720.0	3085.6
	$\mathbf{n = 20}$	**UB Gap**	1.4%	1.2%	3.3%	2.4%
		LB Gap	-1.4%	-1.2%	-3.3%	-2.4%
		Time (Cut-Gen)	1965.5	1944.4	1955.0	1805.2
		Time (CPLEX)	3600.5	3600.4	3601.1	3600.5
TF = 0.6	$\mathbf{n = 10}$	**UB Gap**	0.0%	0.0%	0.0%	0.0%
		LB Gap	-1.0%	-1.7%	-1.1%	-1.7%
		Time (Cut-Gen)	28.6	27.8	22.1	62.3
		Time (CPLEX)	3.7	15.2	82.7	314.9
	$\mathbf{n = 15}$	**UB Gap**	0.1%	0.3%	0.5%	0.9%
		LB Gap	-1.4%	-2.4%	-1.0%	-1.7%
		Time (Cut-Gen)	351.0	867.4	869.7	1582.9
		Time (CPLEX)	68.3	730.4	3407.5	3195.4
	$\mathbf{n = 20}$	**UB Gap**	2.6%	0.9%	4.7%	5.2%
		LB Gap	-3.4%	-1.8%	-4.9%	-5.6%
		Time (Cut-Gen)	3417.8	3413.9	3491.1	3282.3
		Time (CPLEX)	2088.5	1219.7	3600.1	3600.1

Table 4.3: Effectiveness of the cut-generation algorithm under TWT performance measure ($\alpha = 0.90$).

and retrieve the best upper and lower bounds that could be achieved within this limit. For the Lagrangian subproblems of the cut-generation algorithm, we imposed an initial time limit of $n/2$ seconds for TWT measure and $n/4$ seconds for TCT and TWCT measures.

For a given instance, the upper bound gap is computed with respect to the optimal solution if it is available. Otherwise, the best known lower bound is determined by taking the maximum of our lower bound and the best lower bound retrieved from CPLEX, and the optimality gap is computed with respect to this lower bound. Similarly, lower bound gap is computed with respect to the optimal solution. If the optimal solution is not available, the difference between the lower bound obtained by the cut-generation algorithm and the best known upper bound is divided to the best known lower bound that is achieved. The formulas for gap calculations, when the knowledge of optimality does not exist, are presented below.

$$\text{UB Gap} = \frac{UB_{cut-gen} - \max(LB_{cplex}, LB_{cut-gen})}{\max(LB_{cplex}, LB_{cut-gen})},$$
$$\text{LB Gap} = \frac{LB_{cut-gen} - \min(UB_{cplex}, UB_{cut-gen})}{\max(LB_{cplex}, LB_{cut-gen})}.$$

where UB and LB corresponds to the best feasible solution and best lower bound obtained from the solution routine described in the subscript. For each n, the first two rows in Tables 4.1-4.3 specify the average upper and lower bounds on the gaps ("UB Gap" and "LB Gap") for the cut-generation algorithm. The last two rows presents the average elapsed times in seconds for the cut-generation algorithm and for CPLEX.

Several conclusions may be drawn from Tables 4.1-4.3. First, it is obvious that the algorithm does not work well for the objectives TCT and TWCT. Although the gaps are fairly small, CPLEX is able to solve the instances to optimality within very small amounts of time. This could be partially attributed to the fact that a small number of constraints is required to model the scheduling objectives of minimizing TCT and TWCT. On the other hand, in order to minimize TWT, we require additional sets of constraints and variables. Further, incorporating VaR makes the problem more difficult, causing CPLEX to perform poorly. We observe this in Table 4.3. Solving VaR-TWT is very time consuming even for small $|N|$ as the number of scenarios grows. In fact many instances are failed to be solved to optimality after $|N| = 15$ and $|S| = 100$. On the other hand, the quality of bounds provided by the cut-generation algorithm is quite high for all $|N|$, although there is a slight decrease when $|N| = 20$ which could be attributed to the premature termination due to the time limit. Here, we underline the capability of our algorithm to handle large number

of scenarios while CPLEX fails to provide sufficient lower bounds. Since we achieved the best results for TWT, we will further continue our analysis using only this objective.

Our second analysis compares the upper and lower bound gaps of the cut-generation algorithm and CPLEX where the optimal solution is unknown, i.e. CPLEX was aborted due to time limit. In order to make a fair comparison, we also exclude the cases where the initial bounding scheme of the cut-generation algorithm is sufficient to determine the optimal solution. In order to obtain a large sample, we ignored the differences on the tardiness factors and aggregated the data.

		$\lvert S\rvert$			
		50	100	150	200
$\mathbf{n = 15}$	**UB Gap (Cut-Gen)**	-	8.20%	2.91%	2.29%
	LB Gap (Cut-Gen)	-	-7.93%	-3.42%	-2.45%
	UB Gap (CPLEX)	-	7.93%	2.66%	2.21%
	LB Gap (CPLEX)	-	-11.66%	-10.68%	-30.14%
$\mathbf{n = 20}$	**UB Gap (Cut-Gen)**	3.75%	2.07%	5.06%	4.74%
	LB Gap (Cut-Gen)	-3.58%	-1.89%	-5.16%	-4.96%
	UB Gap (CPLEX)	3.60%	1.89%	4.98%	4.94%
	LB Gap (CPLEX)	-23.91%	-31.53%	-23.06%	-47.22%

Table 4.4: Effectiveness of the cut-generation algorithm under TWT performance measure only for cases where CPLEX terminated due to time limit ($\alpha = 0.90$).

Table 4.4 supports the fact that CPLEX is unable to provide sufficient lower bounds when $|N| \geq 15$. In fact, when $|N| = 20$, it returns a trivial lower bound of 0 for several instances. On the contrary, the lower bound gap of cut-generation algorithm is in general less than 5%, and the maximum gap is below 9%. Furthermore, the upper bound gaps of both algorithms is quite close to each other which suggests that the cut-generation algorithm can achieve sufficient upper bounds in a reasonable amount of time when the number of scenarios is high.

Our third study was carried out in order to analyze the effect of the number of scenarios to the VaR that is achieved. More specifically, we would like to know at least how many scenarios should be generated so that the VaR remains unchanged henceforth. We assumed that true value of VaR is approximated the best when $|S| = 200$, since we could at most solve up to this many scenarios. In this study, we focused on 5 instances with $|N| = 10$ and $|S| = 200$, where we iteratively decremented the number of scenarios and reoptimized the problem. We computed the gap of VaR with respect to the case where $|S| = 200$. Figure 4.1 displays our results where the number of scenarios were varied

between 25 and 200.

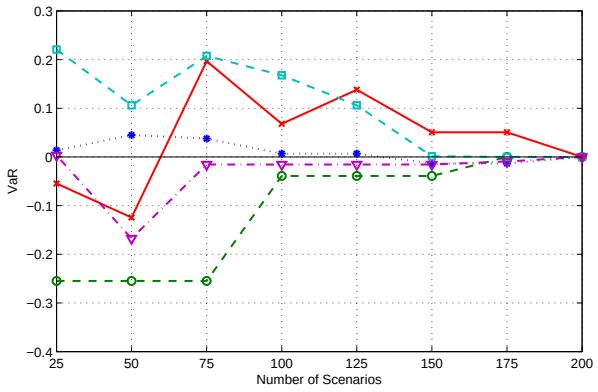


Figure 4.1: Gap of VaR with respect to the number of scenarios for 5 instances. Gap is computed according to the best available approximation of the true VaR which is $|S| = 200$ in our case.

We immediately notice that for small numbers of scenarios, we observe high fluctuations in the VaRs. However, these fluctuations tend to decrease when the number of scenarios is increased. Nevertheless, even when $|S| = 175$, we observe a gap as large as 5% which suggests that $|S|$ should be increased even further to approximate the true value of VaR better.

In our fourth study, we present the effect of parallel programming on the solution times of the cut-generation algorithm. We have used the instances with $TF = 0.4$ and $n < 20$ in order to keep the analysis concise. The results are presented in Table 4.5.

		Nb. of Subproblem Threads		
		1	2	4
$\mathbf{n = 10}$	$\mathbf{\|S\| = 50}$	31.53	21.69	16.65
	$\mathbf{\|S\| = 100}$	13.21	9.13	7.59
	$\mathbf{\|S\| = 150}$	113.38	75.56	53.17
	$\mathbf{\|S\| = 200}$	44.96	33.36	26.40
$\mathbf{n = 15}$	$\mathbf{\|S\| = 50}$	1536.19	1199.57	789.81
	$\mathbf{\|S\| = 100}$	824.05	1009.13	815.89
	$\mathbf{\|S\| = 150}$	1298.33	717.96	674.83
	$\mathbf{\|S\| = 200}$	2081.81	2134.62	2140.70

Table 4.5: The effect of using multiple threads for subproblems on the average elapsed times of cut-generation algorithm.

For $n = 10$, we clearly observe the effect of parallelization where the solution times

decrease on average 30% with 2 threads and 46% with 4 threads, when compared to the serial algorithm. Notice that the solution times do not halve when the number of threads is doubled, which would correspond to an ideal 100% efficiency. Efficiency is a measure, commonly used in the context of parallel programming in order to examine the utilization of multiple threads. We measure efficiency using the formula below:

$$\text{Efficency} = \frac{\text{Elapsed time using a single thread}}{\text{Elapsed time using N threads } \times \text{ N}}$$

We observe on average 72% and 47% efficiencies for 2 and 4 threads, respectively. The loss on efficiency could be both attributed to internal operations other than solving subproblems, and to our parallelization strategy. Remember that we tackle the subproblems in batches and continue with the next batch if only all the members of the current batch is completely solved. Notice that if we increase the size of the batches we would be more likely to keep the processors idle, therefore observe a lower utilization. This is supported by the fact that the efficiency decreases with increasing numbers of threads.

For $n = 15$, the rates of improvement in average solution times drop down to 10% and 24%, and the average efficiencies decrease to 61% and 37% for 2 and 4 threads, respectively. This decrease in performance can be explained by the difficulty of the VaR-TWT problem even when $n = 15$. The differences between the solution times of the subproblems are more sharp when compared to the subproblems of $n = 10$, leading to more idle processors. Moreover, the algorithm cannot converge to a solution within the time limit for several instances, therefore aborted prematurely. For instance, when $|S| = 200$, either the optimal solution is determined in the initialization or the algorithm terminates due to time limit. As a result, we cannot fully observe the effect of parallelization. Nevertheless, even a small percentage of improvement leads to a significant reduction in total solution times as can be observed in Table 4.5. Further, using multiple threads increases the quality of the final outcome for prematurely terminated instances. This is because a larger number of iterations can be carried out within the same time limit. In conclusion, we claim that parallel programming has a great impact on the computational performance of our algorithm, and carries a great potential.

Finally, in Table 4.6, we give the average number of iterations required for our proposed algorithm to solve the problems. We observe that the number of iterations for larger values of $|N|$ and $|S|$ are higher when compared to the smaller values of these parameters. We note that this is due to the time limit of the algorithm. In general, we do not observe any significant trends between the number of iterations and the rest of the parameters.

		$\|S\|$			
		50	100	150	200
TF = 0.4	$\mathbf{n = 10}$	39.4	18.8	72.0	46.4
	$\mathbf{n = 15}$	44.4	103.2	27.8	30.2
	$\mathbf{n = 20}$	1.6	11.8	9.2	10.0
TF = 0.6	$\mathbf{n = 10}$	48.2	64.2	33.6	52.2
	$\mathbf{n = 15}$	63.0	41.2	34.7	47.8
	$\mathbf{n = 20}$	24.8	38.2	15.4	19.8

Table 4.6: The number of iterations spend by the cut-generation algorithm to solve the problems.

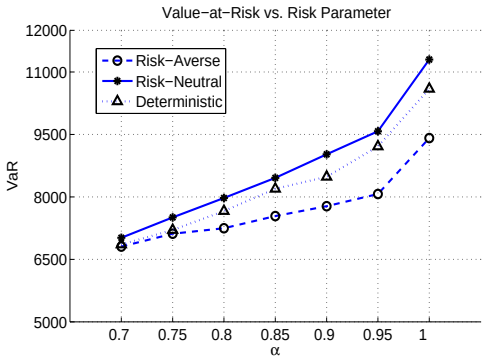
4.3 Value of the Risk-Averse Model

The value of a risk-averse solution depends on the relative performance of the corresponding deterministic and risk-neutral solutions as a function of the risk appetite. Therefore, in this part, we benchmark VaR-$f(x)$ against corresponding deterministic and risk-averse models as the risk parameter α is varied. In this section, we focused on only TWT problem, whereas similar results could be obtained for any other performance measure. The deterministic counterpart of VaR-TWT problem is the conventional single-machine TWT problem, in which all processing times take on their expected values; that is, we have $p_j = \bar{p}_j = \sum_{s \in S} \pi^s p_j^s$. In the risk-neutral version of our problem, we minimize the expected TWT by solving the following formulation:

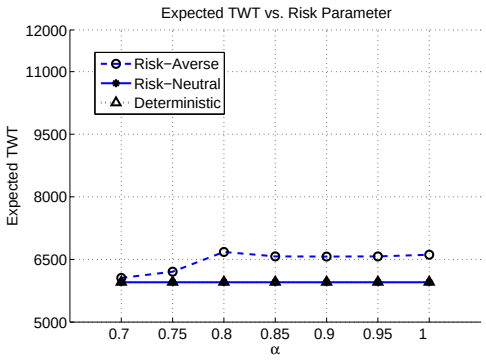
$$\begin{aligned} \min \quad & \sum_{j=1}^{n} w_j \sum_{s \in S} \pi^s T_j^s \qquad (4.1) \\ \text{subject to} \quad & (2.5) - (2.10). \end{aligned}$$

In Figure 4.2, we zoom into two instances from Table 4.3 to illustrate how the VaR changes as α is varied. For this data set we obtain risk-averse solutions without sacrificing much from the expected TWT as α increases.

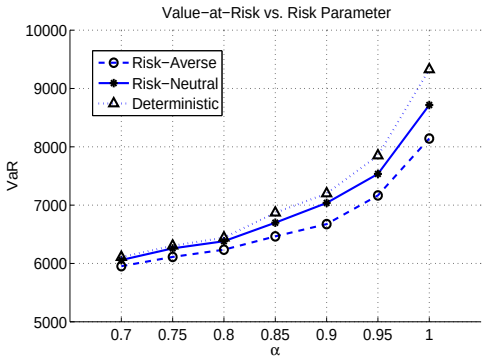
Finally, we use the additional data that is described in Section 4.1 which have higher variability in the processing times. All scenarios are assumed to be equally likely. A total of 10 instances for $n = 10, 15$ and TF=0.6 are solved by the risk-neutral model and cut-



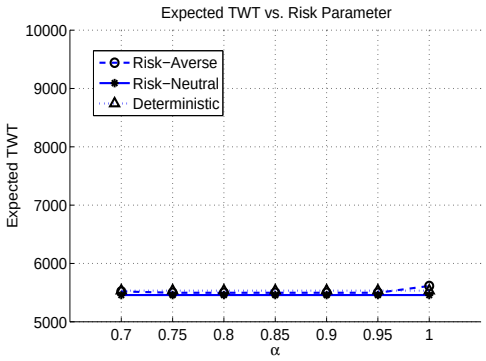
(a) $n = 10, \mid S \mid = 200$, Instance I



(b) $n = 10, \mid S \mid = 200$, Instance I



(c) $n = 10, \mid S \mid = 200$, Instance II



(d) $n = 10, \mid S \mid = 200$, Instance II

Figure 4.2: Comparison of the risk-averse model to its deterministic and risk-neutral counterparts.

generation algorithm for $\alpha = 0.90$. For these 10 instances, the entries in Table 4.7 indicate the relative decrease in VaR and the relative increase in the expected TWT for the solution of the cut-generation algorithm in comparison to that of the risk-neutral model. In this table, we refer to the data with CV$= 0.16$ as Data Set 1, and CV$= 0.26$ as Data Set 2. The risk-averse solution exhibits significant improvements over the risk-neutral solution, albeit at times at the expense of the expected TWT to hedge against the uncertainty.

	$\mathbf{n = 10}$				$\mathbf{n = 15}$			
	DataSet 1		**DataSet 2**		**DataSet 1**		**DataSet 2**	
$\mid \mathbf{S} \mid$	θ	**E(TWT)**	θ	**E(TWT)**	θ	**E(TWT)**	θ	**E(TWT)**
50	-3.15%	3.02%	-5.03%	5.05%	-2.23%	0.70%	-3.89%	2.40%
100	-5.49%	6.90%	-4.75%	5.99%	-0.32%	1.60%	-3.78%	2.89%
150	-5.17%	0.71%	-0.62%	0.62%	-15.83%	0.00%	-1.51%	2.33%
200	-13.81%	10.35%	-2.28%	5.55%	-16.18%	3.54%	-5.41%	7.71%

Table 4.7: The risk-averse model (cut-generation algorithm) versus the risk-neutral model ($\alpha = 0.90$).

Chapter 5

Conclusion and Future Work

In this thesis, we modeled the problem of minimizing VaR in the single-machine scheduling problems under the presence of uncertainty and illustrated the value of the proposed risk-averse model. To solve our single-stage risk-averse stochastic model, we adapted the Lagrangian based solution strategy of Carøe and Schultz (1999) which was originally developed for two-stage stochastic programming models. Furthermore, we considered a variable splitting based Lagrangian relaxation algorithm for minimizing Value-at-Risk. To the extent of our knowledge, this is the first in the stochastic programming literature, and can be applied to a wide variety of settings other than machine scheduling. We proposed solution methods in order to solve the Lagrangian subproblems, and introduced a promising cut-generation algorithm to solve the Lagrangian dual problem. In this study, we focused on minimizing completion time, weighted completion time, and weighted tardiness. However, a wider variety of objectives could also be examined. An extension of our work would be incorporating non-regular objectives such as minimizing earliness-tardiness. Moreover, the solution approach we have implemented could be embedded into a branch and bound algorithm. As a result, we could be able to solve the problem of minimizing VaR to optimality. Finally, additional risk measures, such as the conditional-value-at-risk, could be considered instead of VaR, leading to more choices for the preferences of a decision maker.

In order to improve the performance of our solution procedure, a ranking assignments algorithm could be used. This algorithm successively solves assignment problems in order to generate K-many solutions with increasing costs to the original assignment problem. It was first proposed by Murty (1968), and later its computational performance and complexity was improved by Pascoal et al. (2003). Notice that the use of such an

algorithm could allow us to compute the optimal solution of more subproblems without requiring solving integer programs. By generating and evaluating the K suboptimal solutions of the assignment problem, the trade-off between the term $(\pi^s + \mu^s)\theta^s$ and $\sum_{j \in N} \sum_{k \in N} u_{jk} H^s x_{jk}$ in (3.22) could be resolved easily. Even if the K suboptimal assignments are insufficient to resolve the trade-off, we can still make use of the best available solution and append it as a suboptimal cut to the restricted master problem. Unfortunately, such a strategy is only valid when the subproblems are modeled using APDF. This is because to the best of our knowledge a ranking based algorithm for the linear ordering problem does not exist.

A final improvement on the computational performance would be regarding the parallelization strategy that we follow. In our current algorithm, we solve the subproblems in batches. Further, in order to move on to the next batch we wait all the subproblems in the current batch to be completely solved. Due to this waiting, we are not able to utilize the processors in full efficiency. Although we try to balance the load of the processors, we still observe idle processors and excess waiting times. A suggestion would be gathering all the subproblems to a pool, then dequeuing subproblems whenever a processor becomes idle. Such a strategy will considerably enhance the utilization of the processors. Since the subproblems consume the largest portion of the total effort we spend on our solution method, such a modification will surely result in a more efficient and a faster algorithm.

Bibliography

Ahmed, S., Tawarmalani, M., and Sahinidis, N. V. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100:355–377.

Alouloua, M. A. and Croce, F. D. (2008). Complexity of single machine scheduling problems under scenario-based uncertainty. *Operations Research Letters*, 36:338–342.

Artzner, P., Delbaen, F., Eber, J. M., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9:203–227.

Birge, J., Donohue, C. J., Holmes, D. F., and Svintsiski, O. G. (1996). A parallel implementation of nested decomposition algorithm for multistage stochastic linear programs. *Mathematical Programming*, 75:327–352.

Birge, J. and Louveaux, F. (1997). *Introduction to stochastic programming*. Springer, New York.

Bulbul, K., Kaminsky, P., and Yano, C. (2007). Preemption in single machine earliness/tardiness scheduling. *Journal of Scheduling*, 10(4-5):271–292.

Carøe, C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83:451–464.

Carøe, C. C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37 45.

Charnes, A., Cooper, W., and Symonds, G. (1958). Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, 4:235–263.

Chvátal, V. (1983). *Linear programming*. W. H. Freeman and Company, New York.

Daniels, R. and Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single stage production. *Management Science*, 41:363–376.

de Farias, JR, I. R., Zhao, H., and Zhao, M. (2010). A family of inequalities valid for the robust single machine scheduling polyhedron. *Computers and Operations Research*, 37:1610–1614.

de Panne, C. V. and Popp, W. (1963). Minimum cost cattle feed under probabilistic constraints. *Management Science*, 9:405–430.

Dentcheva, D. (2006). *Probabilistic and Randomized Methods for Design under Uncertainty*, chapter Optimization Models with Probabilistic Constraints. Springer-Verlag, London. editor: Calafiore, G. and Dabbene, F.

Frangioni, A. and Gendron, B. (2010). A Stabilized Structured Dantzig- Wolfe Decomposition Method. Technical report, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation.

Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Gutjahr, W. J., Hellmayr, A., and Pflug, G. C. (1999). Optimal stochastic single-machine-tardiness scheduling by stochastic branch-and-bound. *European Journal of Operational Research*, 117:396–413.

Hiriart-Urruty, J. B. and Lemaréchal, C. (1993). *Convex Analysis and Minimization Algorithms*. Springer, Berlin.

Jörnsten, K. O., Näsberg, M., and Smeds, P. A. (1985). Variable splitting a new lagrangean relaxation approach to some mathematical programming models. Technical Report Department of Mathematics Report LiTH-MAT-R-85-04, Linköping Institute of Technology, Sweden.

Kallehauge, B., Larsen, J., and G.., M. O. B. (2006). Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33:1464–1487.

Kasperski, A. (2005). Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion. *Operations Research Letters*, 33:431–436.

Kataoka, S. (1963). A stochastic programming model. *Econometrica*, 31:181–196.

Keha, A. B., Khowala, K., and Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers and Industrial Engineering*, 56(1):357–367.

Klein Haneveld, W. K. and van der Vlerk, M. H. (1999). Stochastic integer programming:general models and algorithms. *Annals of Operations Research*, 85:39–57.

Laporte, G. and Louveaux, F. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.

Larsen, N., Mausser, H., and Uryasev, S. (2002). *Financial Engineering, e-commerce and Supply Chain*, chapter Algorithms for Optimization of Value-at-Risk, pages 129–157. Kluwer Academic Publishers, Berlin. P. Pardalos and V. K. Tsitsiringos (Eds.).

Lawler, E. L. (1977). A 'pseudo-polynomial' time algorithm for sequencing jobsto minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342.

Lehmann, E. (1955). Ordered families of distributions. *Annals of Mathematical Statistics*, 26:399–419.

Lenstra, J., Rinnooy Kan, A., and Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362.

Linderoth, J. and Wright, S. (2003). Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24:207–250.

Louveaux, F. and Schultz, R. (2003). *Stochastic Programming, Handbooks in Operations Research and Management Science 10*. Elsevier, Amsterdam. editors: A. Ruszczyński and A. Shapiro.

Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60.

Muller, A. and Stoyan, D. (2002). *Comparison methods for stochastic models and risks*. Wiley, New York.

Murty, K. G. (1968). An Algorithm for Ranking all the Assignments in Order of Increasing Cost. *Operations Research*, 16(3):682–687.

Neame, P., Boland, N., and Ralph, D. (2000). An outer approximate subdifferential method for piecewise affine optimization. *Mathematical Programming*, 86:57–86.

Nemhauser, G. L. and Savelsbergh, M. W. P. (1992). *Combinatorial Optimization: New Frontiers in the Theory and Practice, volume 82 of NATO ASI Series F: Computer and System Sciences*, chapter A cutting plane algorithm for the single machine scheduling problem with release times, pages 63–84. Springer, Berlin. M. Akgül, H. Hamacher, and S. Tufekci (Eds.).

Ogryczak, W. and Ruszczyński, A. (1999). From stochastic dominance to mean-risk models: semideviations as risk measures. *European Journal of Operational Research*, 116:33–50.

Ogryczak, W. and Ruszczyński, A. (2002). Dual stochastic dominance and related mean-risks models. *SIAM Journal of Optimization*, 13(2):60–78.

Pascoal, M., Captivo, M. E., and Clímaco, J. a. (2003). A note on a new variant of Murty's ranking assignments algorithm. *4OR*, 255(1):243–255.

Pinedo, M. (1995). *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, Englewood Cliffs,NJ.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition.

Potts, C. and van Wassenhove, L. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 1(5):177–181.

Prékopa, A. (1995). *Stochastic Programming*. Kluwer Academic, Dordrecht, Boston.

Rafael, M. and Reinelt, G. (2011). *Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Springer.

Rockafellar, R. T. and Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16:119–147.

Ruszczyński, A. (1993). Parallel decomposition of multistage stochastic programming problems. *Mathematical Programming*, 58:201–228.

Schultz, R. and Tiedemann, S. (2003). Risk aversion via excess probabilities in stochastic programs with mixed-integer recourse. *SIAM J. on Optimization*, 14(1):115–138.

Schultz, R. and Tiedemann, S. (2006). Conditional value-at-risk in stochastic programs with mixed-integer recourse. *Mathematical Programming*, 105(2):365–386.

Sen, H. (2010). A simple, fast, and effective heuristic for the single-machine total weighted tardiness problem. Master's thesis, Sabancı University, Istanbul, Turkey.

Sen, T., Sulek, J. M., and Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *International Journal of Production Economics*, 83(1):1–12.

Srinivasan, V. (1971). A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Naval Research Logistics Quarterly*, 18(3):317–327.

Tanaka, S., Fujikuma, S., and Araki, M. (2009). An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, 12:575–593.

Van Slyke, R. M. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Wolsey, L. A. (1998). *Integer Programming*. Wiley, USA.

Yang, J. and Yu, G. (2002). On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6:17–33.